Jül-4304

Jülich Supercomputing Centre (JSC)

Konzeption eines Verfahrens zur dynamischen Freischaltung von Transportverbindungen über einen Grid-Scheduler

Thomas Oistrez

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
FORSCHUNGSZENTRUM

Konzeption eines Verfahrens zur dynamischen Freischaltung von Transportverbindungen über einen Grid-Scheduler

Thomas Oistrez

Berichte des Forschungszentrums Jülich; 4304
ISSN 0944-2952
Jülich Supercomputing Centre (JSC)
Jül-4304

Vollständig frei verfügbar im Internet auf dem Jülicher Open Access Server (JUWEL) unter
http://www.fz-juelich.de/zb/juwel

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek, Verlag
D-52425 Jülich · Bundesrepublik Deutschland
☎ 02461 61-5220 · Telefax: 02461 61-6103 · e-mail: zb-publikation@fz-juelich.de

Die vorliegende Masterarbeit wurde in Zusammenarbeit mit dem Forschungszentrum Jülich GmbH, Jülich Supercomputing Centre (JSC), angefertigt.

Diese Masterarbeit wurde betreut von:

Referent: Prof. Dr. rer. nat. V. Sander (Fachhochschule Aachen)

Koreferent: Dipl.-Inf. R. Niederberger (Forschungszentrum Jülich)

Diese Arbeit wurde von mir selbständig angefertigt und verfasst. Es sind keine anderen als die angegebenen Quellen und Hilfsmittel benutzt worden.

Thomas Oistrez
Jülich, 29. Juni 2009

Abstract

The Grid technology was developed to give scientists better and more flexible access to worldwide IT resources. A grid is a distributed system, which makes different IT resources available, even if these are distributed over many administratively independent facilities at different sites. These resources are computing power, storage space and data which are available and shareable for complete work groups in a Grid.

A Grid system has special demands to the underlying networks. Most applications use high amounts of data as well as many small files. Reliable and stable connections are as important as high bandwidth to enable efficient working. Because of the complex organization and technical infrastructure, several problems arise considering the safety and performance of file transfers. Existing solutions are either very slow or they need a very insecure configuration of the participating organization network.

In this master thesis, a new method to communicate between grid components is presented. It should be a secure and easily configurable concept that integrates well in the architecture of a grid system. Therefore, active configuration of access rights by a firewall agent is used. The agent is part of the grid. Firewall configurations should be resources that the grid knows about and that can be reserved like other resources. The method can easily be managed and monitored because it is tightly integrated into the grid. The active configuration enables any kind of direct connection. Thus, most disadvantages of the existing methods are corrected.

Zusammenfassung

Grid-Systeme wurden entwickelt, um Wissenschaftlern einen besseren und flexibleren Zugang zu den weltweit vorhandenen IT-Ressourcen zu verschaffen. Ein Grid ist ein verteiltes System, welches den Nutzern verschiedene IT-Ressourcen zugänglich macht, die auf mehrere, administrativ unabhängige Einrichtungen an unterschiedlichen Standorten verteilt sind. Diese Ressourcen sind z.B. Rechenleistung, Speicherkapazität und Daten, welche für ganze Arbeitsgruppen in einem Grid verfügbar gemacht und untereinander ausgetauscht werden können.

Ein Grid-System hat einige besondere Eigenschaften und Anforderungen an die zugrunde liegenden Netzwerke. Die meisten Anwendungen arbeiten sowohl mit großen Datenmengen, als auch mit vielen kleinen Dateien. Zuverlässige und stabile Verbindungen sind dazu ebenso wichtig wie hohe Transferraten, um ein effizientes Arbeiten zu ermöglichen. Durch die komplexen organisatorischen und technischen Strukturen ergeben sich aber Probleme in Bezug auf Sicherheit und Geschwindigkeit von Dateitransfers. Existierende Lösungen sind entweder sehr langsam, oder sie erfordern eine sehr unsichere Konfiguration der Netzwerke bei den teilnehmenden Organisationen.

In dieser Arbeit wird ein neues Verfahren zur Kommunikation zwischen Grid-Komponenten vorgestellt. Es soll ein sicheres und leicht zu konfigurierendes Konzept entwickelt werden, das sich gut mit der Architektur eines Grid vereinbaren lässt. Dazu werden aktive Freischaltungen durch einen Firewall-Agenten verwendet. Dieser ist Teil des Grids. Die Verbindungsfreigabe soll eine Ressource darstellen, derer sich das Grid bewusst ist und die, wie andere Ressourcen im Grid, reserviert werden kann. Ein solches Verfahren ist durch die enge Einbindung in das Grid gut zu verwalten und zu überwachen. Die aktive Konfiguration macht jede Art von direkter Verbindung möglich. Damit sind die meisten Nachteile der existierenden Lösungen bereinigt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Grid-Systeme wurden entwickelt, um Wissenschaftlern einen besseren und flexibleren Zugang zu den weltweit vorhandenen IT-Ressourcen zu verschaffen. Zur Lösung von Problemen in Wissenschaft und Forschung sind Computer heute unbedingt notwendig. Viele Entwicklungen sind nur durch Simulationen und Berechnungen auf Computersystemen möglich. Die dabei benötigte Rechenleistung steigt ständig an. Auch ein Supercomputer kann in vielen Fällen nicht alle Anforderungen eines Forschungsprojektes erfüllen. Es entstand daher bei Wissenschaftlern der Wunsch, weltweit vorhandene und auf viele Organisationen aufgeteilte Rechenleistung gemeinsam nutzen zu können. Im Umfeld der universitären und staatlichen Forschung wurde dazu das Konzept des Grid-Computing entwickelt, dass die punktuell vorhandenen Ressourcen der weltweiten Forschung zugänglich machen sollte. Mittlerweile setzen neben der Grundlagenforschung auch kommerzielle Unternehmen, z.B. aus der Pharmaindustrie, dem Wirtschafts- und Finanzwesen oder der Automobilindustrie auf die Möglichkeiten, die das Grid-Computing bietet. Durch diese kommerzielle Nutzung entstanden weitere Anforderungen und Bedingungen an die Grids, wie in der folgenden Beschreibung ersichtlich wird.

1.1 Was ist ein Grid?

Ein Grid ist ein verteiltes System, welches den Nutzern verschiedene Ressourcen zugänglich macht. Diese Ressourcen sind auf mehrere, administrativ unabhängige Einrichtungen an unterschiedlichen Standorten verteilt. Dabei handelt es sich um Rechenleistung, Speicherkapazität, Datenverbindungen, Software und Daten. Ein Grid weist dabei einige Besonderheiten und Unterschiede auf, im Vergleich zu anderen verteilten Konzepten wie z.B. dem Cluster computing, Peer-to-Peer Netzen, oder Distributed Computing Projekten wie SETI@Home [16] oder Folding@Home [17]. Diese Unterschiede bestehen mehrheitlich in der Organisation und Verwaltung eines Grids in Virtuellen Organisationen.

Für Forschungsprojekte, an denen mehrere Organisationen beteiligt sind, ist es üblich solche virtuellen Organisationen zu bilden. Das bedeutet, dass mehrere reale Organisationen (Unternehmen oder Forschungseinrichtungen) sich zu einer einzigen virtuellen Organisation (VO) zusammenschließen. Die verschiedenen Mitglieder einer VO können sich an weltweit verteilten Standorten befinden und aus verschiedenen Unternehmen, Branchen bzw. Einrichtungen stammen. Einige reale Organisationen bringen Ressourcen in das Grid mit ein, während andere keine eigenen Ressourcen haben, sondern nur als Nutzer auftreten. Dabei kann ein Unternehmen komplett der VO beitreten oder es nehmen nur eine Arbeitsgruppe oder einzelne Personen aus einer realen Organisation teil. Eine VO hat durch zentrale Authentifizierung und Autorisierung eine eigene, unabhängige Zugriffs- und Sicherheitspolitik, welche von den Richtlinien der realen Organisationen abweichen kann. Die Rolle einer Person in der VO muss nicht der Rolle in ihrer realen Organisation entsprechen. Das Grid in Form einer Grid-Middleware auf den teilnehmenden Rechnern stellt die technische Realisierung einer solchen VO dar. Abbildung 1.1 stammt aus [4] und verdeutlicht nochmals den Aufbau virtueller Organisationen.

Eine wichtige Eigenschaft, die ein Grid-System haben muss, ist demnach die Ortstransparenz. Ortstransparenz bedeutet in einem Grid-System, dass es keine Rolle spielt, wo sich der Benutzer bzw. ein bestimmtes Gerät befindet. Das gleiche gilt auch für die Speichersysteme, auf die immer gleich, unabhängig von ihrem weltweiten Standort, schreibend sowie lesend zugegriffen werden kann. Ein Rechner oder Speicher im Nachbarraum sollte genau so ange-

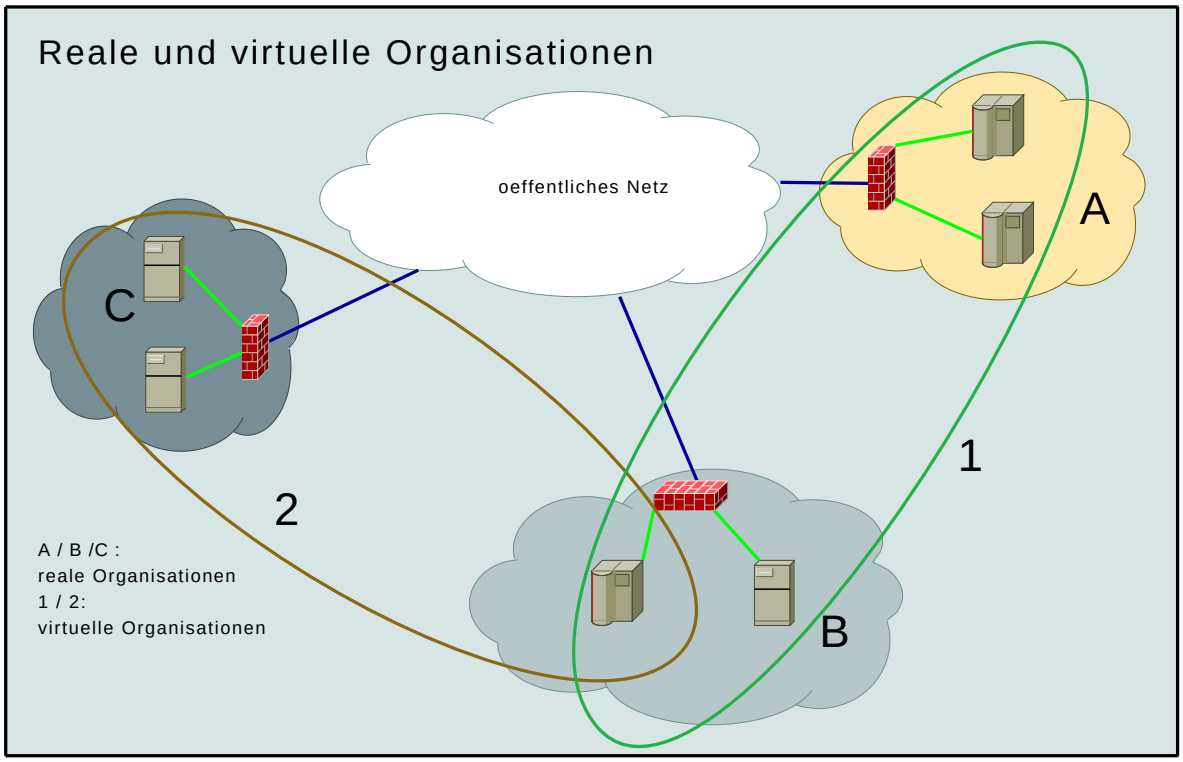

Abbildung 1.1: Virtuelle Organisationen

sprochen werden können wie ein geographisch weit entferntes System. Nur so kann ein Grid effizient und intuitiv bedient werden, wie es auch bei einem lokalen System der Fall wäre.

Zur Besonderheit eines Grids im Vergleich zu vielen lokalen Systemen gehört die dynamische Bereitstellung von Ressourcen. So können jederzeit einzelne Systeme in das Grid integriert oder aus diesem entfernt werden, um das Gesamtsystem auf einem aktuellen, leistungsfähigen Stand zu halten. Durch diese Möglichkeit können neu erworbene Komponenten einfach und schnell in ein bestehendes Grid integriert werden ohne den laufenden Betrieb zu unterbrechen. Damit ändern sich Anzahl und Standort der einzelnen Komponenten im Grid ständig.

Ein weiterer wichtiger Aspekt im Grid ist die System- und Plattformunabhängigkeit. Sie ist notwendig, damit jedes zur Verfügung stehende System in das Grid integriert werden kann, unabhängig von dessen Hardware-Architektur und dem verwendeten Betriebssystem. Vom Arbeitsplatzrechner bis zum Supercomputer müssen alle Systeme gleich benutzbar sein. Damit ist sichergestellt, dass alle zur Verfügung stehenden Ressourcen in einem einzigen Grid je VO zusammengefasst werden können und es nicht aus technischen Gründen zu einer Partitionierung des Grids bzw. zu Insellösungen kommt.

Gängige Grid-Systeme nutzen zur technischen Realisierung dieser Plattformunabhängigkeit wie bereits erwähnt sog. Middleware. Dies sind Softwarekomponenten, welche die bisher genannten Eigenschaften implementieren. Eine Middleware basiert auf der jeweiligen Hard- und Software-Architektur und versteckt deren spezielle Eigenschaften vor den Nutzern und anderen Systemen. So kann auf alle Systeme auf die gleiche Weise zugegriffen werden. Midd-

leware muss somit leicht zu portieren sein, um sie für die eingesetzten Architekturen anpassen zu können.

Die Middleware übernimmt auch die Kommunikation zwischen den einzelnen Rechnern in einem Grid, so dass der Nutzer keine Informationen über Standort und Art der einzelnen Komponenten des Grids haben muss. Außerdem werden grundlegende Verwaltungsaufgaben wie Authentifizierung und Autorisierung von der Grid-Middleware übernommen. Auch erweiterte Funktionen wie die Abrechnung der verbrauchten Ressourcen zwischen den realen Organisationen können von einer Middleware übernommen werden. Abbildung 1.2 zeigt einen möglichen Aufbau eines Grids aus Hardware- und Software-Komponenten, wobei die logische Unterteilung des Grids in drei Bereiche dargestellt ist. Diese drei Ebenen bestehen jeweils aus beliebig vielen Geräten, die alle einem logischen Grid angehören.

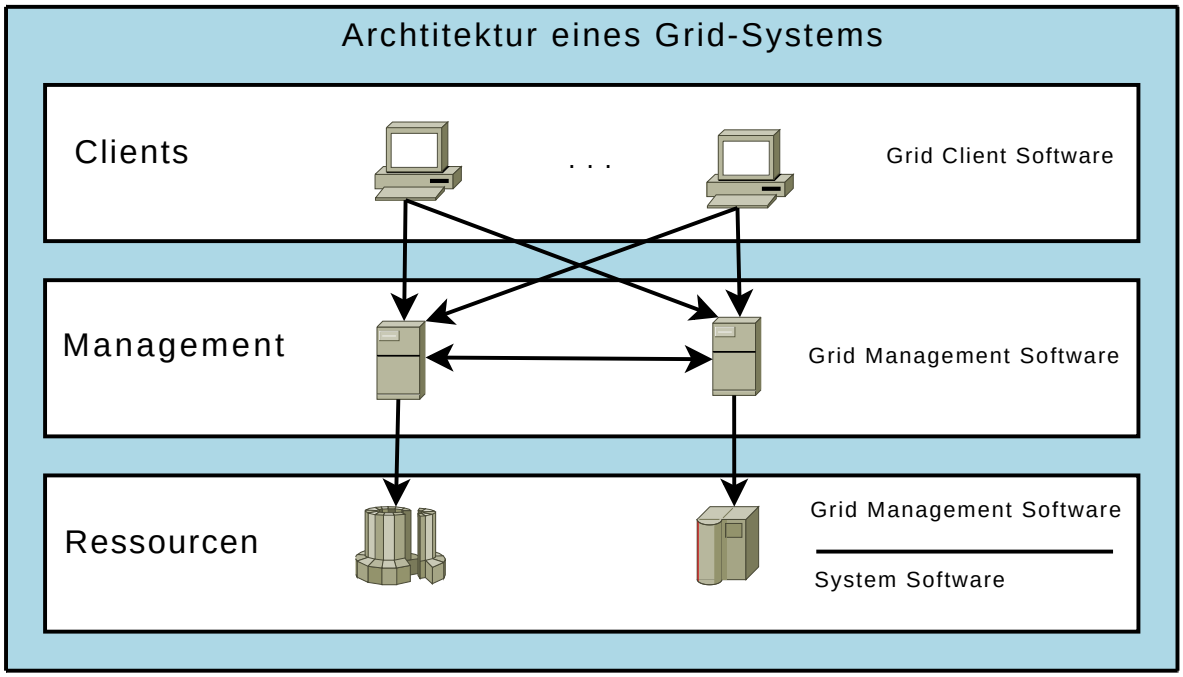

Abbildung 1.2: Aufbau eines Grids

Die Rechenleistung der im Grid zusammengefassten Rechnersysteme wird allen Nutzern von der Middleware zugänglich gemacht, so dass deren anstehende Berechnungen immer auf dem System ausgeführt werden können, welches zur Zeit die größten Kapazitäten frei hat oder in speziellen Fällen am Besten dafür geeignet ist. Kriterien für die Auswahl eines Systems sind neben der momentanen Auslastung die Rechenleistung und die Größe des Hauptspeichers, aber auch das Vorhandensein spezieller Bibliotheken und benötigter Software sowie die Bandbreite und Qualität der Netzwerkverbindung zu diesem System. So kann von den Besitzern der Ressourcen eine optimale Auslastung erreicht werden. Den Nutzern wird dadurch eine schnellstmögliche Bearbeitung der anstehenden Aufgaben garantiert. Ein wichtiges Feature im Grid stellt die Möglichkeit dar, automatisierte Workflows zu erstellen, die mehrere, voneinander abhängige, Berechnungen auf verschiedenen Systemen beinhalten und ohne weiteres Zutun des Benutzers ablaufen.

Bei wissenschaftlich-technischen Modellberechnungen fallen große Datenmengen an, die

auf allen an der Berechnung beteiligten Systemen zur Verfügung stehen müssen, so dass eine große Menge an persistentem Speicher notwendig wäre. In einem Grid werden Speicher aber ebenso flexibel und ortsunabhängig genutzt wie die integrierten Rechner. Als Teil des Gesamtsystems ist jeder persistente Speicher von allen Systemen und für alle Nutzer zugänglich. Diese einzelnen Speicherkapazitäten können z.B. in Form von ganzen RAID-Systemen und Magnetspeichern in das Grid mit eingebracht werden. So kann die insgesamt zur Verfügung stehende Speicherkapazität flexibel und ortsunabhängig zwischen den Nutzern aufgeteilt werden. Sie fließt zusammen mit der Rechenleistung und anderen Kriterien in die Bewertung der Systeme für anstehende Jobs ein.

Zudem ist ein einfaches Austauschen von Daten und berechneten Ergebnissen sinnvoll, um die Zusammenarbeit der weltweit verteilten Organisationen und Arbeitsgruppen zu verbessern. In einem sogenannten DataGrid können daher Daten, die bei Berechnungen im Grid angefallen sind, zwischen den Nutzern geteilt und öffentlich zugänglich gemacht werden. Dieses Prinzip wird oft mit den oben beschriebenen Konzepten kombiniert.

1.2 Datenkommunikation in Grids

Ein Grid-System, wie es im vorherigen Abschnitt beschrieben wurde, hat einige besondere Eigenschaften und Anforderungen an die zugrunde liegenden Netzwerke, die näher betrachtet werden sollten. Die meisten Anwendungen benötigen für ihre Berechnungen große Datenmengen, die sowohl in sehr großen Dateien als auch in vielen kleinen Dateien vorliegen können. Eine ständige Erreichbarkeit sowie zuverlässige und stabile Kommunikationswege zwischen allen Systemen im Grid sind zum Transport dieser Dateien unverzichtbar. Die Client-Software jedes Nutzers muss mit allen Rechnern und Speichern im Grid Daten austauschen können. Dabei wird nicht immer direkt kommuniziert, sondern auch mithilfe von Gateway-Rechner, die Nachrichten und Daten weiterleiten. Aber auch die Rechner lesen und schreiben Daten auf allen Speichern und für Workflows ist Kommunikation zwischen mehreren Rechner nötig. Ebenso wichtig wie die Zuverlässigkeit der Kommunikation ist eine hohe Transferrate. Eine hohe Auslastung der Systeme kann nur erreicht werden, wenn die zur Berechnung notwendigen Daten rechtzeitig auf den Rechnern vorliegen. Langsamer Datenaustausch kann also alle Prozesse im Grid ausbremsen. Ein einzelnes langsames Teilnetz wird dabei schnell zu einem Flaschenhals für das gesamte Grid. Im Extremfall überwiegt die benötigte Zeit für Datentransfers gegenüber der eigentlichen Rechenzeit, was dem Gedanken der hohen Ressourcenauslastung und des effizienten Arbeitens widerspricht.

Außer der benötigten Bandbreite und der Zuverlässigkeit spielt auch die Ortstransparenz eine wichtige Rolle für die Netzwerke. Der Datenaustausch erfolgt zwischen sämtlichen Komponenten des Netzes. Die einzelnen Systeme sind dabei nicht unbedingt in der selben realen Organisation angesiedelt, sondern über die ganze VO, also weltweit verteilt. Datenverbindungen müssen über mehrere autonome Netze hinweg aufgebaut werden, unabhängig von den Netztopologien und der Verwaltungsstruktur der Netze.

Insgesamt kann ein Grid also nur optimal funktionieren wenn jederzeit schnelle und zuverlässige Kommunikation zwischen allen Komponenten möglich ist. Die Vorraussetzungen, die diese Anforderungen an die Middleware und die zu Grunde liegenden Kommunikationsnetze stellt, werden im nächsten Kapitel genauer betrachtet.

Kapitel 2

Problemstellung

2.1 Problemstellung

Die Funktionen eines Grids wurden im letzten Kapitel beschrieben und die Anforderungen an die Datenkommunikation erläutert. Nun soll genauer auf die Netzwerksicherheit aus Sicht des Betreibers eingegangen werden.

2.1.1 Anforderungen des Netzwerkbetreibers

Der Betreiber eines Netzwerks, also eine reale Organisation bzw. die zuständige Abteilung innerhalb der Organisation, hat genaue Anforderungen an die Sicherheitseigenschaften des Netzwerks. Diese werden üblicherweise durch die folgenden Begriffe definiert:

Verfügbarkeit Die Verfügbarkeit eines Systems wird allgemein über die Formel

$$Verf\ddot{u}gbarkeit = \frac{Gesamtzeit - Gesamtausfallzeit}{Gesamtzeit} * 100\%$$

ausgedrückt. Sie beschreibt also, welchen Anteil die Zeit, in der man das System nutzen kann, an der Gesamtzeit hat. Die Gesamtzeit entspricht dabei grundsätzlich dem gesamten betrachteten Zeitraum. Angegeben wird die Verfügbarkeit hier in einem Prozentwert, der sich direkt aus der Formel ergibt.

Eine weiteres Kriterium für Verfügbarkeit ist die Kennzahl "Mean Time between Failure" (MTBF). Sie beschreibt die durchschnittliche Betriebsdauer zwischen zwei Ausfällen. Da die beiden Kriterien die Verfügbarkeit auf verschiedene Weise definieren, werden sie oft kombiniert. Der Betreiber sichert seinen Kunden meist eine hohe Verfügbarkeit (übliche Werte liegen über 95%) sowie eine möglichst lange MTBF zu.

Die wesentliche Maßnahme zur Erhöhung bzw. Erreichung der nötigen Verfügbarkeit ist Redundanz. Die Aufgaben ausfallender Systeme werden von Ersatzsystemen übernommen, die ständig in Bereitschaft sind.

Integrität Der Schutzbedarf für die im Datennetz transportierten Informationen ergibt sich unter anderem aus der Notwendigkeit der Integrität der Informationen. Es ist Ziel des Betreibers, das der Inhalt der Datenpakete sich im Netz nicht verändert. Sowohl Veränderungen durch Fehler in den Netzwerkkomponenten, als auch von Menschen verursachte Manipulationen sind zu vermeiden. Dabei können Manipulationen unbeabsichtigt oder beabsichtigt sein. Für fast alle Anwendungen in einem Datennetz ist die Integrität der Daten zwingend erforderlich. Durch Prüfsummen zu den verwendeten Daten versucht man, verletzte Integrität zu erkennen, um die Daten anschließend erneut zu versenden.

Vertraulichkeit Viele Anwendungen in einem Datennetz sind auf ein gewisses Maß an Vertraulichkeit ihrer Daten angewiesen. Vertraulichkeit beschreibt die Eigenschaft der Daten, dass sie nur für einen bestimmten Kreis von Personen/Geräten gedacht sind. Ihr Inhalt muss vor allen anderen verborgen bleiben. Die übliche Maßnahme zur Gewährleistung der Vertraulichkeit stellt die Verschlüsselung der Daten dar.

Widerspruchsfreiheit Anstatt von Widerspruchsfreiheit wird auch von Nichtzurückweisbarkeit gesprochen. Es geht dabei darum, dass keiner der beiden Kommunikationspartner die Kommunikation nachträglich leugnen kann. Es soll beweisbar sein, dass die

Nachricht wirklich vom Sender stammt und wirklich beim beabsichtigten Empfänger angekommen ist. Hierzu werden z.B. Zertifikate zur Unterschrift und Verschlüsselung eingesetzt.

Sicherheit ist also maßgeblich auf diesen Begriffen aufgebaut. Wenn die Anforderungen nicht eingehalten werden, dann kann das für Unternehmen zu schweren Problemen führen. Mangelnde Verfügbarkeit führt zu eingeschränkter Produktivität. Nicht eingehaltene Integrität hat möglicherweise Fehler im Geschäftsablauf zur Folge. Fehlende Vertraulichkeit führt möglicherweise zum Verlust von Betriebsgeheimnissen. Diese Probleme können sich alle in finanziellen Verlusten äußern.

Zur Sicherung eines Netzwerkes gehören auch Überlegungen zu den möglichen Angriffsszenarien. Dabei kann grundsätzlich zwischen Angriffen unterschieden werden, die beabsichtigt sind, und solchen, die unbeabsichtigt entstehen. Unbeabsichtigte Angriffe, z.B. von eigenen Mitarbeitern, passieren oft von einem System innerhalb das eigenen Netzes aus. Auch davor ist das Netz z.B. durch Einschränkung der Rechte Einzelner zu schützen. Angriffe auf die Netzsicherheit zielen oft auf die Störung der Integrität und der Widerspruchsfreiheit, z.B. durch Spoofing, ab. Spoofing beschreibt die Manipulation von Datenpaketen, so dass sie einem anderen Absender zugeordnet werden. Bei Man-in-the-Middle Attacken fängt der Angreifer Pakete ab, verändert sie gegebenenfalls, und leitet sie an den eigentlichen Empfänger weiter. Dies gefährdet die Vertraulichkeit und die Integrität.

Zum Schutz vor Angriffen auf die Systeme im Netzwerk dient im Wesentlichen die Kontrolle des Datenverkehrs durch Firewalls. Diese trennen Bereiche mit verschiedenen Sicherheitsanforderungen voneinander. Sie befinden sich z.B. an den Grenzen des Netzes, also an den Anbindungen zu externen Netzen, aber auch innerhalb des eigenen Netzes, um Bereiche mit besonders sensiblen Systemen zu schützen. Firewalls kontrollieren den Datenverkehr und unterbinden gefährliche Kommunikation. Die Entscheidung über die Gefährlichkeit wird aufgrund eines Regelwerks getroffen, das vom Netzwerkadministrator vorgegeben ist. So sollen Angriffe von außen unterbunden werden, z.B. indem Kommunikationen mit Geräten unterbunden werden, die als gefährlich bekannt sind. Zudem ist es oft üblich nur Verbindungen zuzulassen, die von eigenen Geräten initiiert wurden. Wenn ein System im Inneren des Netzwerks vom Angreifer genutzt wird, dann ist es schwerer, Gegenmaßnahmen zu treffen, da eine Einschränkung der internen Kommunikation meistens eine eingeschränkte Produktivität zur Folge hat.

2.1.2 Anforderungen des Grid-Benutzers

Im letzten Kapitel wurden bereits grundsätzliche Anforderungen eines Grids an die Datenkommunikation beschrieben. Nun soll genauer auf die technischen Konsequenzen dieser Anforderungen eingegangen werden.

Eine wichtige Forderung an die Netzwerke ist die Erreichbarkeit jedes Systems, das am Grid teil nimmt. Von jedem System im Grid aus sollte jedes andere ansprechbar sein. Da auch Systeme anderer Organisationen Teil des Grids sind, muss also Kommunikation mit externen Systemen erlaubt sein. Der Verbindungsaufbau kann dabei in beide Richtungen stattfinden. Außerdem waren kurze Latenzen und hohe Übertragungsgeschwindigkeiten gefordert. Dies kann nur optimal erfüllt werden, wenn direkte Verbindungen aufgebaut werden. Es sind also kurze Wege nötig. Umleitungen und zusätzliche Prüfung wären hinderlich. Da das Grid optimal ausgelastet werden soll, müssen solche Verbindungen jederzeit auch spontan möglich

sein. Eine zeitliche Begrenzung ist nicht möglich. Zu beachten ist außerdem, dass im Grid sehr viele solcher Verbindungen gleichzeitig aktiv sein können. Dabei werden viele verschiedene Protokolle auf Transport- und Anwendungsebene genutzt.

2.1.3 Gegenüberstellung

Betrachtet man die beiden oben beschriebenen Positionen, dann ist leicht ersichtlich, dass sich die Anforderungen beider Seiten schwer vereinbaren lassen. Das Grid funktioniert nur optimal, wenn ein uneingeschränkter und effizienter Datenaustuasch zwischen allen Systemen gewährleistet ist. Zudem sollen die vorhandenen Ressourcen allen Nutzern dynamisch zuteilbar sein.

Die Partner in einem Grid können aber durchaus auch direkte Konkurrenten auf dem Markt sein. Gemeinsamer Zugriff mit einem Konkurrenten auf ein System stellt offensichtlich eine Gefahr für die Vertraulichkeit der eigenen Daten dar.

Die vielen Systeme im Grid machen eine sehr hohe Zahl an Firewall-Regeln für die Kommunikation nötig. Durch das dynamische Hinzufügen und Entfernen von Geräten ändert sich das Regelwerk ständig. Das macht die Handhabung und Pflege des Regelwerks aufwändig und fehleranfällig. Gröbere und freizügigere Regeln machen das Netz zwar nutzbar, hinterlassen aber ein dauerhaft niedrigeres Sicherheitsniveau.

Insgesamt schließen sich optimal funktionierende Grid-Systeme und hohe Anforderungen an die Sicherheit des eigenen Netzwerks also auf den ersten Blick aus. Es sind Konzepte gefragt, die die Ansprüche beider Seiten so gut wie möglich vereinbaren.

2.2 Existierende Lösungen

In zur Zeit eingesetzter Grid-Software sind verschiedene Verfahren zum Austausch von Dateien verfügbar. Im Folgenden soll ein Überblick dieser existierenden Lösungen gegeben werden. Jedes Verfahren wird vorgestellt und anhand der Kriterien des letzten Unterkapitels bewertet.

2.2.1 Tunneling

Eine Möglichkeit, die geschilderten Sicherheitsprobleme zu umgehen, stellt das Tunneln der Verbindungen dar. Dabei werden die Verbindungen zwischen zwei Endsystemen nicht direkt hergestellt, sondern z.B. durch einen http- oder ssh-Tunnel geleitet. Die realen Organisationen müssen dazu Tunnel-Server betreiben, die die verschiedenen Verbindungen ähnlich einem Proxi-Server entgegen nehmen und durch eine einzige Verbindung zur anderen Organisation leiten. Dieses Vorgehen reduziert die Zahl der eigenen Systeme, die in direktem Kontakt mit externen Systemen stehen und ermöglicht dadurch ein kleines, immer aktuelles Regelwerk.

Die Konzentration der Verbindungen auf einen zentralen Rechner sowie das Multiplexen der Daten machen dieses Verfahren aber langsamer als direkte Verbindungen. Optionale Verschlüsselung der Datenströme erhöht zwar die Vertraulichkeit, kostet aber ebenfalls Zeit. Zudem erzeugt man mit dem Tunnel-Server einen Single-Point-of-Failure, was die Verfügbarkeit gefährdet, und es ergeben sich Probleme mit eventuell eingesetzen Content-Filtern in den Firewalls. Diese erkennen, dass das eingesetzte Anwendungsprotokoll (z.B. http) keine zum Protokoll passenden Daten transportiert und verwerfen die Datenpakete. Tunnel-Techniken sind daher weniger flexibel und aufwändiger zu realisieren als direkte Verbindungen.

In der verbreiteten Grid Software UNICORE [15] stellen solche Tunnel die Standardverfahren zum Dateitransfer dar. Die ohnehin vorhandenen Gateway-Server dienen as Tunnel-Server und leiten alle Daten entweder als WebService-Aufrufe (ByteIO-Verfahren) oder über zusätzliche HTTP-Verbindungen (BFT-Verfahren) weiter. Dieses Vorgehen ist zwar sehr sicher, liefert aber nicht die vom Benutzer gewünschte Performance.

2.2.2 GridFTP

GridFTP [13] ist ein auf FTP basierendes Dateitransfer-Protokoll. Es überträgt Dateien zwischen Client und Server oder zwischen zwei Servern über TCP-Verbindungen. Dazu werden Kontrollverbindungen und davon unabhängige Datenverbindungen aufgebaut. GridFTP wurde speziell für den Einsatz in Grid-Umgebungen entwickelt und unterscheidet sich daher in einigen Punkten von FTP. Die Kontrollverbindung wird verschlüsselt, während bei FTP alle Informationen im Klartext übertragen wurden. Dies erhöht die Vertraulichkeit. Außerdem können für einen Dateitransfer nicht nur eine, sondern mehrere Datenverbindungen geöffnet werden. Die zu übertragende Datei wird dann auf die parallelen Verbindungen aufgeteilt. Das erhöht die Durchsatzrate im Vergleich zu normalem FTP erheblich und lässt die Übertragungsrate weniger stark einbrechen wenn ein Datenpaket einer Verbindung verloren geht. GridFTP beherrscht „commandline pipelining“ und kann so mehrere Aktionen hintereinander ausführen ohne neue TCP-Verbindungen aufbauen zu müssen. Bekannte Grid-Middleware wie z.B. Globus nutzt GridFTP als Verfahren für Dateitransfers.

GridFTP erreicht eine hohe Netzauslastung, indem es mehrere TCP-Ströme gleichzeitig nutzt und die Daten gleichmäßig darauf aufteilt. Es kann insgesamt als effizient und schnell bezeichnet werden, die Verwendung von vielen Verbindungen geht allerdings auf Kosten der Fairness zum restlichen Datenverkehr im Netz. Ferner sinkt die Effizienz bei einer zu großen Anzahl von parallelen Verbindungen, da der steigende Verwaltungsaufwand den Durchsatz behindert.

Der Einsatz von GridFTP erzwingt aber eine sehr unsichere Netzkonfiguration. Da die Kontrollverbindung im Gegensatz zu FTP verschlüsselt ist, können Firewalls den Datenverkehr nicht mitlesen und somit auch die Datenverbindungen keiner Kontrollverbindung zuordnen. Während bei FTP die Firewalls Datenverbindungen bezogen auf die übermittelten Kontrollinformationen dynamisch an der Firewall freischalten können, ist das bei GridFTP nicht möglich. Es müssen für alle Systeme, die GridFTP nutzen, permanent viele ein- und ausgehende Verbindungen möglich sein, auch wenn GridFTP gerade nicht aktiv ist. Dies widerspricht den beschriebenen Sicherheitsanforderungen.

2.2.3 UDP-Hole-Punching

UDP hole punching ist ein Verfahren zur dynamischen Konfiguration von Firewalls. Es setzt nur die Erlaubnis für ausgehenden Datenverkehr voraus, was bei den meisten Sicherheitsrichtlinien ohnehin der Fall ist. Es ist dann möglich, direkten Datenaustausch zwischen zwei Endpunkten herzustellen, indem den Firewalls auf beiden Seiten ein von innen initiierter Datenaustausch vorgespielt wird. Das Verfahren gilt als ausreichend sicher und schnell, funktioniert aber nur mit dem UDP Protokoll. Daher ist bei TCP basierten Verbindung immer eine Anpassung des gesamten Verfahrens nötig. Eine detaillierte Beschreibung zum UDP hole punching liefert [4].

2.3 Das neue Verfahren

In dieser Arbeit soll ein neues Verfahren vorgestellt werden. Es soll versucht werden, ein sicheres und leicht zu konfigurierendes Konzept zu finden, das sich gut mit der Architektur eines Grid vereinbaren lässt. Dazu werden keine indirekten Freischaltungen wie beim UDP-Hole-Punching verwendet, sondern aktive Freischaltungen durch einen Firewall-Agenten. Dieser ist Teil des Grids. Die Verbindungsfreigabe soll eine Ressource darstellen, derer sich das Grid bewusst ist und die, wie andere Ressourcen im Grid, reserviert werden kann.

Ein solches Verfahren wäre durch die enge Einbindung in das Grid gut zu verwalten und zu überwachen. Die aktive Konfiguration würde jede Art von direkter Verbindung möglich machen. Damit wären die meisten Nachteile der existierenden Lösungen bereinigt.

Die grundlegenden Konzepte für ein solches Verfahren werden in den nächsten beiden Kapiteln vorgestellt. Dabei wird auf allgemeine Standards und Methoden eingegangen, wie sie heute schon in Grid-Middleware eingesetzt werden, oder sich zur Zeit in Entwicklung befinden. Die Gesamtarchitektur des Verfahrens wir dann in Kapitel 6 ausführlich behandelt.

Kapitel 3

Grid Scheduling

3.1 Einführung

Bevor auf die besonderen Eigenschaften im Grid Umfeld eingegangen wird, soll hier der Begriff des Schedulings näher erläutert werden. Unter Scheduling versteht man das Erstellen eines Ablaufplans für anstehende Aufgaben. Außer den Aufgaben werden die vorhandenen Ressourcen betrachtet, die zur Erledigung der Aufgaben nötig sind. Schedulingverfahren werden in der Betriebswirtschaftlehre und in der Informatik angewendet. Dabei werden in der Informatik meist die auf einem Computer laufenden Prozesse als die Aufgaben betrachtet, und die zur Verfügung stehenden Prozessoren sowie der Arbeitsspeicher stellen die Ressourcen dar.

Es ist zwischen zwei Arten von Scheduling zu unterscheiden, dem präemptiven und dem nicht-präemptiven Scheduling. Bei einem nicht-präemptiven Verfahren behält ein Prozess die ihm zugewiesenen Ressourcen so lange, bis er die Ressourcen freiwillig wieder abgibt oder der Prozess beendet ist. Ein präemptives Vefahren kann dem Prozess die Ressourcen allerdings jederzeit wieder entziehen. Der Prozess pausiert dann in seinem aktuellen Zustand, bis er wieder Ressourcen zugeteilt bekommt.

Für beide Arten des Scheduling gelten allgemein die folgenden Optimierungskriterien zur Verteilung von Ressourcen an die Prozesse:

Durchsatz Optimaler Durchsatz bedeutet, dass möglichst viele Prozesse in möglichst kurzer Zeit abgearbeitet werden. Alternativ zum Durchsatz kann auch die Auslastung der Ressourcen maximiert werden, man spricht dann von optimaler Effizienz. Die beiden Ansätze sind äquivalent.

Fairness Die Ressourcen müssen gerecht verteilt werden. Das bedeutet, dass kein Prozess übermäßig vernachlässigt werden darf, so das er keine Ressourcen mehr bekommt.

Termineinhaltung Manche Prozesse haben eine sogenannte Deadline. Das heißt, sie müssen bis zu einem bestimmten Zeitpunkt vollständig abgearbeitet sein. Ob solche Termine eingehalten werden oder nicht wird über die Echtzeitfähigkeit definiert. Harte Echtzeitfähigkeit garantiert, das alle Deadlines eingehalten werden. Weiche Echtzeitfähigkeit hält die Deadline zumindest annähernd ein, ein Best-Efford-Ansatz gibt gar keine Garantien.

In einem PC dient das Scheduling der Prozesse hauptsächlich dazu, beliebig viele Aufgaben quasi gleichzeitig laufen zu lassen, wobei der Nutzer einen Eindruck von Parallelität und Echzeitverhalten gewinnen soll. Dazu ist präemptives Scheduling nötig. Im Supercomputing, und somit auch im Grid-Umfeld, sind die Anforderungen dagegen anders. Ein Prozess, hier auch *Job* genannt, wird definiert und an das Schedulingsystem weitergereicht. Interaktivität ist normalerweise nicht nötig. Der Nutzer wird informiert, wenn der Prozess vollständig abgearbeitet wurde. Daher wird hier nicht-präemptives Scheduling verwendet. Auch wenn das Betriebssystem des Supercomputer präemptives Scheduling beherrscht, verhält es sich wie nicht-präemptives Scheduling, weil ein Prozess immer mindestens einen Rechenkern und dazugehörigen Speicher fest zugeordnet bekommt, den er sich mit keinem anderen Prozess teilen muss. So müssen einem Prozess nie Ressourcen entzogen werden.

3.2 Service Level Agreements

Service Level Agreements (SLA) werden in Dienstleistungsverträgen zwischen einem Auftraggeber und einem Dienstleister genutzt. Die SLA enthält die genaue Beschreibung des Umfangs der Dienstleistung und der Dienstgüte. Der Sinn einer solchen speziellen Vereinbarung liegt in den transparenten Kontrollmöglichkeiten. Zugesicherte Leistungseigenschaften wie etwa Umfang der zu erbringenden Leistungen oder maximale Zeitdauer bis zur Bearbeitung werden in einer SLA genau beschrieben. Dazu beschreiben SLAs vor allem das Service Level, auch Dienstgüte oder Quality of Service genannt. SLAs wurden ursprünglich für IT-Dienstleistungen eingeführt, sind inzwischen aber in allen Dienstleistungsbranchen üblich.

Eine Besonderheit bei SLAs ist, dass der Dienstleister von vorneherein für alle relevanten Dienstleistungsparameter verschiedene Servicelevel anbietet, die der Dienstnehmer einsehen kann. Darauf basierend kann der Dienstnehmer einen individuellen Service auswählen. Kriterien hierfür sind zum Beispiel die angegebenen Kosten, die von Level zu Level unterschiedlich sind.

Angewendet auf den Bereich des Scheduling bieten SLAs eine allgemeine Möglichkeit, Ressourcen zu reservieren. Auf einem HPC System stellt der Betreiber des Systems bzw. das von ihm betriebene Scheduling-System den Dienstleister dar. Der Nutzer des Grids bzw. der von ihm erstellte Job ist der Auftraggeber. Der Scheduler nennt die zur Verfügung stehenden Ressourcen, und der Nutzer wählt abhängig von seinen Anforderungen und seinem Budget den gewünschten Leistungsumfang, also z.B. die Anzahl der Prozessoren und die Dauer der Nutzung. Dieses Konzept ist integraler Bestandteil des im Folgenden beschriebenen Grid-Scheduling Konzeptes.

3.3 Grid-weites Scheduling

Das bisher beschriebene Scheduling ist auf ein einzelnes System begrenzt. Der Scheduler hat dabei die volle Kontrolle über alle zur Verfügung stehenden Ressourcen. Im Grid-Umfeld ist aber noch eine weitere Form des Scheduling möglich. Bei den in Kapitel 1.2 beschriebenen Workflows ist diese neue Form nötig. In einem Grid gibt es mehrere technisch unabhängige Systeme, die für einen großen und komplizierten Job kombiniert werden können. Jedes dieser Systeme betreibt seinen eigenen, unabhängigen Scheduler für die Ressourcen seines Systems. Die Ausführung eines Workflows läuft daher im einfachsten Fall so ab, das ein Teil-Job nach dem anderen reserviert und ausgeführt wird. So können natürlich erhebliche Wartezeiten zwischen den Teilschritten entstehen, weil auf jedem System neu reserviert und anschließend auf die Zuteilung gewartet werden muss.

Bei bestimmten Ressourcen ist ein einzelnes Scheduling gar nicht möglich. So können Bandbreitenreservierungen auf den genutzten Netzwerken auch als Ressource behandelt werden. Diese müssen sofort zur Verfügung stehen, wenn ein Transfer ansteht. Wartezeiten könnten die beteiligten Systeme blockieren, was auf jeden Fall zu vermeiden ist.

Eine Lösung für dieses Problem stellt ein übergeordneter Grid-Scheduler dar. Diesem sollten allen im Grid verwendeten Ressourcen bekannt sein. Der Grid-Scheduler reserviert nicht direkt die Ressourcen, sondern wendet sich an den jeweiligen lokalen Scheduler. Dort erhält er eine Reservierung für eine bestimmte Ressource zu einem festgelegten Zeitpunkt in der Zukunft und für eine festgelegte Dauer.

Da für einen einzigen Workflow bei verschiedenen lokalen Schedulern angefragt werden

muss und die Ressourcen in einer bestimmten Reihenfolge benötigt werden ist ein mehrstufiges Reservierungsverfahren nötig. Der Grid-Scheduler muss wiederholt Angebote der lokalen Scheduler einholen und abgleichen, bis eine geeignete Angebotskette vorliegt. Erst dann werden die Reservierungen fest vereinbart.

Es gibt verschiedene Ansätze zur Durchführung dieser mehrstufigen, dynamischen Verhandlungen. In [1] wurde ein erstes Verhandlungsverfahren implementiert. Spätere Ansätze versuchen, auf verbreiteten und standardisierten Verfahren aufzubauen. In [2] und [3] werden Verfahren vorgestellt, die SLAs und ein WebService-basiertes Reservierungsprotokoll, den WS-Agreement Standard (WS-AG), nutzen. Die Verhandlungsstrategien werden durch die beschriebenen Eigenschaften der SLAs gut unterstützt. Der WS-AG Standard wird im folgenden Abschnitt vorgestellt.

3.4 Der WS-Agreement Standard

3.4.1 WebServices

Der WS-Agreement Standard basiert auf der WebService-Technologie. WebServices nutzen die üblichen Techniken und Protokolle des World Wide Web (WWW) für Machine-Machine-Kommunikation anstatt der im WWW üblichen Mensch-Machine-Kommunikation. Die Funktionalität von WebServices entspricht dabei in etwa der eines Remote Procedure Call (RPC) von einem Client (Dienstnehmer) an einen Server (Dienstanbieter). Im Unterschied zu anderen RPC-Techniken sind WebServices durch die standardisierten und weit verbreiteten Protokolle plattform-, betriebssystem- und programmiersprachenunabhängig. Die wichtigsten Komponenten des WebService-Standards werden im Folgenden vorgestellt:

HTTP Das HyperText Transfer Protocol (HTTP) ist ein einfaches, Text-basiertes Anfrage-Antwort-Protokoll. Es wird von Clients im WWW verwendet, um Anfragen an einen Server zu senden, der mit dem angefragten Dokument antwortet. WebServices versenden über HTTP die RPC-Aufrufe und -Rückgabewerte.

SOAP Das SOAP-Protokoll beschreibt die mittels HTTP transportierten Nachrichten zwischen Dienstnehmer und Dienstanbieter. Es definiert einen Rahmen für die eigentlichen Daten und Konventionen für die RPC-Aufrufe. Der anwendungsspezifische Inhalt der Nachrichten wird von SOAP nicht vorgegeben. SOAP-Nachrichten basieren wie ihr Inhalt auf XML.

XML Die eXtensible Markup Language (XML) ist eine allgemeine Auszeichnungssprache. Sie ist keinem bestimmten Einsatzzweck zugeordnet. Über sogenannte XML-Schemata kann die erlaubte Struktur von XML-Daten genauer spezifiziert und eingeschränkt werden, um sie an einen konkreten Einsatzzweck anzupassen. Solche Schemata beschreiben z.B. das Format der SOAP-Nachrichten und der im Folgenden beschriebenen WSDL-Dateien.

WSDL Die WebService Definition Language (WSDL) ist ein XML basiertes Dateiformat. WSDL Dateien beschreiben den genauen Umfang eines WebServices. Dazu gehören die zur Verfügung gestellten Methoden und die mit diesen Methoden auszutauschenden Dokumente (die Parameter). Diese Dokumente sind wiederum XML basiert.

UDDI Der Verzeichnisdienst UDDI (Universal Description, Discovery and Integration) hält Informationen über die im Netz angebotenen WebServices in Form von WSDL-Dateien bereit. Die Dienstanbieter registrieren ihre Dienste im Verzeichnis, und die Dienstnehmer können im Verzeichnis nach geeigneten Diensten suchen. Es gibt globale Verzeichnisse, die weltweit von allen Dienstanbietern genuzt werden können. Im Grid-Umfeld ist es üblich, das jedes Grid sein eigenes Verzeichnis betreibt.

Der WebService-Standard macht keine Aussagen über die Funktionalität der zur Verfügung gestellten Methoden oder über die Struktur und den Inhalt der ausgetauschten Dokumente. Dies wird für jeden konkreten Einsatzzweck neu definiert. Für einige immer wiederkehrende Aufgaben sind aber in weiteren Spezifikationen bereits genaue Definitionen gemacht worden. Die Spezifikationen bauen teilweise aufeinander auf und werden üblicherweise mit WS-* bezeichnet, wobei * den Anwendungszeck beschreibt. Bekannte Beispiele hierfür sind WS-Addressing, WS-Discovery und der im Folgenden beschriebene WS-Agreement Standard. Abbildung 3.1 faßt die Komponenten und Zusammenhänge in der WebService-Architektur nochmals zusammen.

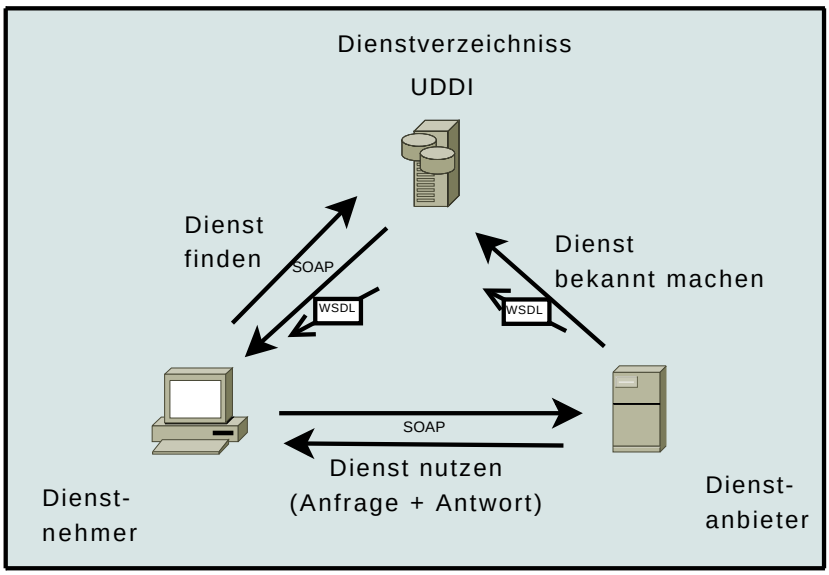

Abbildung 3.1: Übersicht WebServices

3.4.2 WS-Agreement

Der OGF-Standard WS-Agreement wurde von der Arbeitsgruppe *Grid Resource Allocation Agreement Protocol (GRAAP)* des Open Grid Forum erarbeitet. Ziel war es, Standardverfahren für die Vereinbarung (eng. „agreement“) von SLAs über Grid-Ressourcen zu schaffen. Dazu zählt die Definition der bereitgestellten WebServices (Ports) und der Dokumente, die als Parameter ausgetauscht werden.

Zur einfachen Verwendung in möglichst vielen konkreten Einsatzbereichen erfüllt der Standard die wichtigsten Voraussetzungen. Er läßt Struktur und Inhalt der Dienstbezeichnungen offen, so dass jede Art von Dienst beschrieben werden kann. Auch die genauen Bedingun-

gen, z.B. Kriterien zur Dienstgüte, unter denen ein Dienst angeboten wird können beliebig definiert werden.

Dienstnehmer und Diensterbringer sind nicht fest der anfragenden bzw. antwortenden Rolle zugeteilt. Beide können ein Agreement als sogenannter Initiator erstellen und vorschlagen. Die Gegenseite, die das Agreement dann annehmen muss, wird über deren WebService Ports involviert. Welche der beiden Seiten das Agreement-Objekt während seiner Existenz verwaltet ist ebenfalls nicht fest vorgegeben. Die Rollen im Client-Server-Modell können also auf WebService Ebene bzw. bei der Verhandlung über Agreements anders verteilt sein als bei der späteren Bereitstellung bzw. Inanspruchnahme des ausgehandelten Dienstes.

Ebenfalls offen gelassen ist das Verhandlungsverfahren für komplizierte Reservierung mit mehreren Parteien. Einfache Reservierungen zwischen zwei Parteien können aber direkt mit den Mitteln des WS-Agreement Standards erstellt werden. Der Standard setzt auf mehreren anderen WS-* Spezifikationen auf. Dies sind WS-Adressing, WS-ResourceProperties, WS-ResourceLifetime und WS-BaseFaults.

Die WS-Agreement Spezifikation besteht aus drei wesentlichen Teilen: dem Agreement, dem Agreement-Template inklusive der Bedingungen (eng. constraints) sowie den Port-Types und deren Operationen. Zudem werden verschiedene Zustände für die Agreements und Services definiert.

Der Lebenszyklus eines Agreements besteht aus dessen Erstellung, dem Monitoring während des Bestehens und dem Verfall nach der vereinbarten Laufzeit. Auch ein vorzeitiges Aufkündigen ist möglich. Die folgenden Abschnitte orientieren sich an der offiziellen Spezifikation und geben einen Überblick über deren Inhalt. Die vollständige Spezifikation ist in [5] nachzulesen. Dort sind auch die XML Schemata der beschriebenen Datenstrukturen abgedruckt.

Das Agreement

Das Agreement besteht aus drei Teilen: dem Namen, dem Kontext und den Bedingungen (eng. terms). Zudem hat jedes Agreement eine ID.

ID Die ID ist zur genauen Identifikation des Agreement zwingend erforderlich und muss zwischen Initiator und Gegenseite eindeutig sein. So sind verschiedene Versionen oder Modifikationen, die bei erweiterten Verhandlungen auftreten können, eindeutig voneinander unterscheidbar.

Name Der Name ist ein optionaler Wert und kann aus einer beliebigen Zeichenkette bestehen. Inhalt und Bedeutung sind nicht weiter spezifiziert und müssen von den Verhandlungspartnern entsprechend dem konkreten Einsatzzweck gedeutet werden.

Context Dieser Bereich enthält Elemente, die allgemeine Informationen zum Agreement bieten und nicht in den nächsten Abschnitt „Terms“ passen. Dies sind z.B. die Bezeichnungen der beiden Verhandlungspartner oder die Laufzeit des Agreement. Auch die Art des vereinbarten Dienstes kann hier angegeben werden. Zudem sind weitere selbst definierte, vom konkreten Einsatzzweck abhängige Angaben möglich.

Terms Der wichtigste Teil des Agreements sind die genauen Bedingungen (Terms) zum vereinbarten Dienst. Mehrere Bedingungen können zu Gruppen zusammengefasst werden, für die eine der folgenden logischen Operatoren gilt: All (Alle Bedingungen müssen erfüllt sein), OneOrMore (Mindestens eine der Bedingungen muss erfüllt sein) und ExactlyOne (genau eine der Bedingungen muss erfüllt sein). Es gibt zwei Typen von Terms:

die Service Terms und die Guarantee Terms. Erstere beschreiben den Dienst und letztere machen genauere Angaben über die Dienstgüte. Die eigentlichen Parameter des Dienstes sind in den Service Terms festgelegt, die sich wie folgt weiter aufteilen lassen:

service description terms Hier werden die wesentlichen Bedingungen des Agreements festgelegt. Die Service Description Terms (SDTs) sind somit der wichtigste Teil des Agreements. Die SDTs definieren die Funktionalität des vereinbarten Service. Jeder SDT hat einen Namen, einen Verweis auf den behandelten Service, sofern das Agreement mehrere Services umfasst, und einen beschreibenden Teil, dessen Inhalt und Struktur vom konkreten Einsatzzweck abhängt.

service reference Diese Teile verweisen auf andere Dienste, die Gegenstand der Vereinbarung sind. Dazu kann z.B. eine Endpoint Reference (EPR) eines WebService angegeben werden.

service proporties Diese Parameter enthalten offen messbare Kenngrößen des Dienstes, für die ein Wert angegeben werden kann. Beispiele hierfür sind z.B. maximale Antwortzeiten oder Durchsatzraten.

Das Agreement Template

Ein Agreement Template ist eine Schablone, die von einer der beiden Seiten zur Verfügung gestellt wird. Aus ihr kann die Gegenseite einen Vorschlag für ein Agreement erstellen. Ein Agreement Template enhält alle Teile eines Agreements. Es gibt damit die gesamte Struktur vor. Die Werte der eingetragenen Service Terms können als Default-Werte dienen. Zusätzlich enthält ein Template einen Bereich mit Einschränkungen und Vorgaben für die Service Terms (Agreement Creation Constrains). So können z.B. die aktuell zur Verfügung stehenden Ressourcen oder die abrufbaren Funktionen eines Dienstes beschrieben werden. Das spätere Agreement muss sich nicht zwangsläufig an die Struktur und die Einschränkungen des Templates halten. Es ist auch nicht garantiert, dass ein Agreement akzeptiert wird, wenn es alle Einschränkungen erfüllt.

Die einzelnen Einschränkungen, Offer Items genannt, schreiben jeweils das Vorhandensein eines SDTs sowie optional einen gültigen Wertebereich für dessen Inhalt vor. Ein Item besteht aus einem Namen, einem Verweis auf die Position des SDTs im Agreement und dem Wertebereich.

Die Runtime States

WS-Agreement definiert drei verschiedene Zustandsmodelle. Neben dem Agreement können sich die im Agreement beschriebenen SDTs und Guarantee Terms in einem bestimmten Zustand befinden.

Agreement State Ein Agreement durchläuft wärend seiner Existenz verschiedene Phasen. Wenn ein Agreement vom Initiator vorgeschlagen wurde, kann es entweder „Rejected“ (Abgelehnt), „Observed“ (Akzeptiert) oder „Pending“ (noch nicht akzeptiert) sein. Noch nicht akzeptierte Agreements können entweder akzeptiert oder „Terminated“ (Vorzeitig Beendet) werden. Akzeptierte Agreements können vorzeitig beendet oder „Completed“ (Abgeschlossen) werden.

Service Runtime State Jeder SDT befindet sich in genau einem Runtime State. Die Werte sind „Not Ready“, „Ready“ und „Completed“. „Ready“ kann nochmals in „Processing“ und „Idle“ unterteilt werden. Zwischen „Ready“ und „Not Ready“ kann beliebig gewechselt werden. Von „Ready“ kann ein Übergang nach „Completed“ erfolgen. Dieser Zustand wird nicht wieder verlassen.

Guarantee State Die Garantien, die in den Guarantee Terms gegeben werden, können sich in drei verschiedenen Zuständen befinden. Dies sind „Fulfiled“ (erfüllt), „Violated“ (nicht erfüllt) und „Not Determined“. Der letzte Status ist der initiale Wert. Er wird angenommen bevor die Dienstleitung begonnen hat oder zu Zeitpunkten an denen keine messbare Aktivität herrscht. Zwischen den drei Zuständen kann beliebig gewechselt werden.

Die Port-Types

Für das Vorschlagen und Erstellen von Agreements, die Übermittlung von Templates und die Interaktion mit laufenden Agreements zwecks Monitoring definiert der Standard mehrere WebService-Ports. Dabei wird nach dem für WebServices üblichen Factory-Modell vorgegangen, das einen einzelnen Factory-Service bereit stellt, welcher auf Anfrage neue spezialisierte Services erzeugt.

agreement factory Die Agreement-Factory hat zwei Aufgaben. Sie stellt Agreement Templates bereit und nimmt Angebote für Agreements entgegen. Es können beliebig viele Templates als Resource Property vorgehalten werden, die von der Gegenseite mit der Methode GetResourceproperty ausgelesen werden können. Über die Methode CreateAgreement kann ein Initiator einen Vorschlag über ein Agreement einreichen. Die Methode anwortet mit einem Fehler, falls das Agreement abgelehnt wurde oder mit einer EPR auf ein neu erstelltes Agreement Objekt, falls es akzeptiert wurde. Der Initiator kann auch im Vorhinein selbst ein Agreement Objekt anlegen, das dann bei Akzeptanz des Vorschlags genutzt wird. Die EPR zu diesem Objekt muss dann als Teil des Vorschlages mit übermittelt werden.

pending agreement factory Diese Factory hat die selben Aufgaben wie die Agreement Factory. Allerdings kann sich das erstellte Agreement hier auch im Zustand „Pending“ befinden. Eine Entscheidung wird dann erst zu einem späteren Zeitpunkt getroffen. Mitgeteilt wird sie durch ein Agreement Acceptence Objekt auf Seiten des Initators.

agreement acceptance Der Initator eines Agreement kann ein Agreement Acceptence Objekt erstellen und eine EPR auf dieses Objekt zusammen mit einem Agreement Vorschlag an eine Pending Agreement Factory übermitteln. Die Gegenseite nutzt dann die Methoden Accept und Reject dieses Objektes, um seine Entscheidung über das Agreement mitzuteilen.

agreement Ein Agreement Objekt representiert den Zustand eines Agreements. Es bietet die Methode Terminate zur vorzeitigen Aufkündigung des Agreements. Zudem ist der Inhalt des Agreements über mehrere Resource Properties auslesbar.

agreement state Dieses Objekt beinhaltet die bereits beschriebenen Zustände der Terms und des Agreements. Es ist nicht zur alleinigen Verwendung gedacht, sondern wird in konkreten Einsatzzwecken zusammen mit dem Agreement Objekt genutzt.

3.4.3 domain specific Erweiterungen

Der genaue Inhalt der Agreements und Templates hängt stark vom konkreten Einsatzzweck ab. Zur Reservierung von System-Ressourcen sind andere Parameter nötig als zur Reservierung von Netzwerk-Bandbreiten. Zur praktischen Verwendung einer WS-Agreement Implementierung sind also erweiterte, anwendungsbezogene Definitionen für die Inhalte der Dokumente nötig. Dies sind insbesondere die Inhalte der SDTs, die die Eigenschaften der Dienste beschreiben, mit denen sich das Agreement befasst.

Implementierungen tauschen Agreements und Templates in SOAP Nachrichten aus. Dazu werden vom Standard entsprechende XML Dokumente definiert. Die für den Anwendungsfall spezifischen Teile des Dokuments sind offen gelassen. Hier können existierende XML Formate wie z.B. die Job Submission Description Language (JSDL) [6] zur Beschreibung von Rechen-Jobs benutzt werden. Alternativ werden neue Dokumentstrukturen definiert.

Dies ist auch bei der Beschreibung von Verbindungsfreigaben nötig. Daher werden Verbindungsfreigaben im folgenden Kapitel genauer charakterisiert und XML-Strukturen zu ihrer Beschreibung erstellt.

Kapitel 4

Verbindungsfreigaben als Grid Ressource

4.1 Einführung in Firewalls

4.1.1 Die Protokolle TCP und UDP

Heutige Kommunikationsnetze nutzen auf Schicht 3 des ISO-OSI-Modells das Internet Protokoll (IP) zum Austausch von Daten zwischen Endgeräten im Netz. IP ist ein unzuverlässiges, verbindungsloses und paketorientiertes Protokoll, das die zu sendenden Daten in Pakete aufteilt, welche einzeln und unabhängig voneinander verschickt werden. Dabei übernimmt IP keine Garantie dafür, dass alle versendeten Pakete korrekt beim Empfänger ankommen. Sie können verloren gehen, dupliziert werden oder sich gegenseitig überholen. Auf diesem IP Protokoll basieren die beiden Transportprotokolle TCP und UDP. Sie ermöglichen Kommunikation zwischen verschiedenen Prozessen auf den Endgeräten im Netz. Da sich aus den verschiedenen Eigenschaften der beiden Protokolle Unterschiede bei der Behandlung durch Firewalls ergeben, werden sie hier genauer beschrieben.

TCP

TCP dient nicht nur dazu, Prozesse auf den Endgeräten anzusprechen, sondern implementiert noch eine Reihe weiterer Funktionen. TCP nutzt das IP-Protokoll und bietet darauf basierend einen zuverlässigen und verbindungsorientierten Transport von Daten zwischen zwei Prozessen auf verschiedenen Rechnern. Es überwacht die Einhaltung der Reihenfolge und erkennt Duplikate und Paketverlusten. Die dazu nötigen Informationen werden in einem zusätzlichen TCP-Header, welcher in den Nutzdaten des IP-Paketes enthalten ist, übermittelt. Die Zuordnung zu einzelnen Anwendungen auf den jeweiligen Rechnern geschieht durch Port-Nummern, welche einem Prozess exklusiv zugeteilt werden. TCP führt die sogenannten Sequenznummern ein, um Duplikate und in der Reihenfolge vertauschte Pakete, welche auf dem Weg durch das Netz entstehen können, zu erkennen. Jedes einzelne Paket bekommt dabei eine fortlaufende Nummer und ist somit eindeutig identifizierbar. Damit Pakete, welche auf dem Weg zum Ziel verloren gehen, erneut gesendet werden, teilt der Empfänger des Pakets den Erhalt durch ein Antwortpaket, ein sogenanntes ACK[1], mit. Ein ACK kann auch in einem Datenpaket enthalten sein, das in die andere Richtung gesendet wird. Pakete, auf denen nach einer bestimmten Zeit kein ACK erfolgt ist, werden erneut versendet. TCP bietet zusätzlich einen genau definierten Verbindungsauf- und -abbau. In Abbildung 4.1 sind diese beiden Vorgänge dargestellt. Beim Aufbau handelt es sich um einen Three-Way-Handshake. Der Client sendet ein erstes SYN-Paket[2] aus, welches bereits eine erste Sequenznummer enthält. Der Server antwortet durch ein eigenes SYN-Paket mit einer eigenen Sequenznummer, welches auch ein ACK für den Erhalt des ersten SYN-Paketes enthält. Der Client verschickt dann seinerseits nochmal ein ACK an den Server. Dadurch werden technisch gesehen zwei Simplex-Verbindungen etabliert, eine für jede Richtung. Diese beiden Teilverbindungen führen von einander unabhängige Sequenznummern die durch die beiden SYN-Pakete synchronisiert wurden. Der Abbau der Verbindung geschieht einzeln für jede Teilverbindung. Wenn einer der beiden Kommunikationspartner keine weiteren Daten mehr verschicken will, dann schickt er ein FIN-Paket[3]. Dieses wird von der anderen Seite bestätigt. Die TCP-Verbindung gilt als beendet, wenn dieser Abbau in beide Richtungen erfolgt ist. TCP implementiert noch weitere Funktionen wie Stau- und Flusskontrolle, die aber bei der Behandlung durch Firewall keine Rolle spielen.

[1] ACK → „acknowledgment“. dt. Bestätigung
[2] SYN → „synchronize“. dt. Synchronisieren
[3] FIN → „finish “. dt. Abschluss

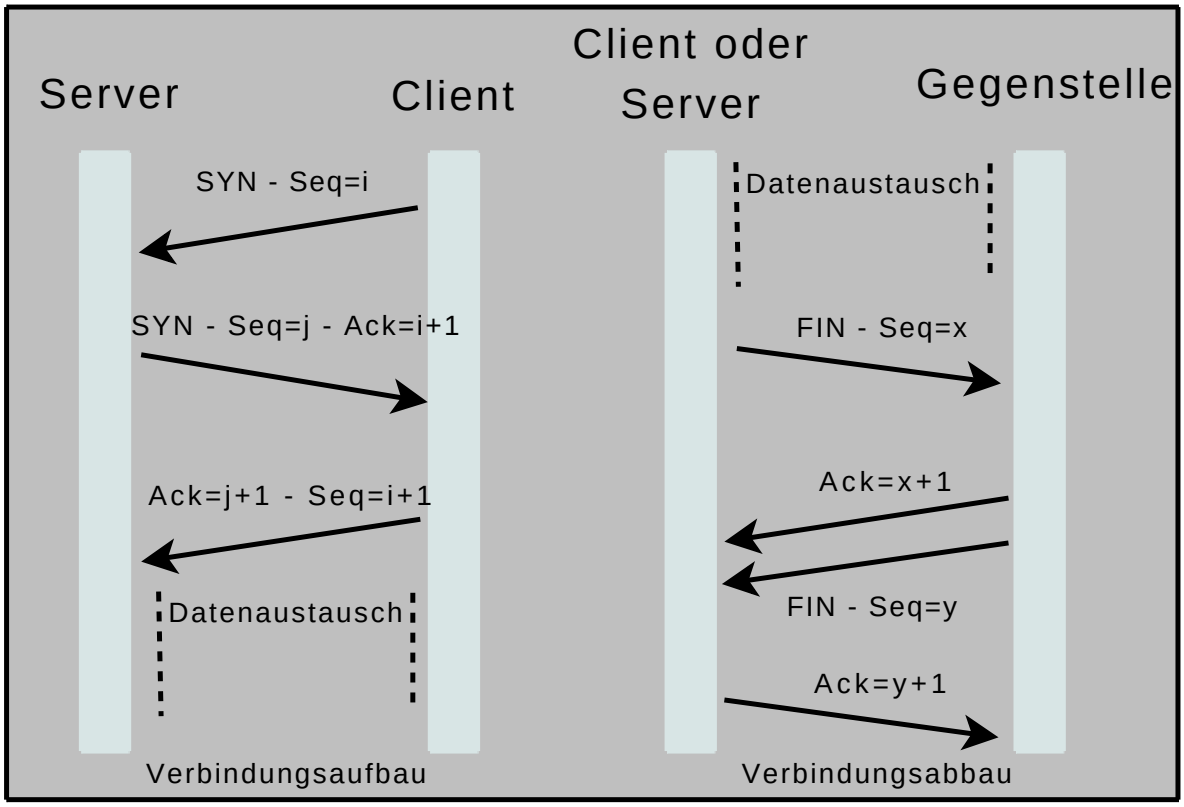

Abbildung 4.1: Three-Way-Handshake

UDP

UDP ist ein wesentlich einfacheres Transportprotokoll. Es verzichtet auf feste Verbindungen. Dadurch werden auch verlorene, duplizierte und vertauschte Pakete nicht erkannt. Jedes UDP Datagramm steht für sich alleine. Die Erweiterungen gegenüber IP beschränken sich im Wesentlichen auf das Einführen von Port-Nummern. Damit ist UDP nur für wenige Anwendungen, wie z.B. das Streamen von Audio- oder Video-Daten, geeignet, bei denen falsche oder fehlende Daten nur zu kleinen akzeptablen Störungen führen. Zudem wird UDP von Anwendungen genutzt, bei denen alle zu übertragenden Daten in ein einziges UDP-Paket passen, wie z.B. DNS und NTP.

4.1.2 Firewalls

Firewalls operieren auf unterschiedlichen Protokoll-Ebenen. Die meisten Firewalls arbeiten heute als sogenannte Stateful-Inspection-Engines. Sie sind sowohl als Software- als auch als Hardware-Komponenten verfügbar. Software-Firewalls werden z.B. als Personal-Firewalls auf Endsystemen eingesetzt, während zur Sicherung ganzer Netze Hardware-Komponenten mit spezieller Software eingesetzt werden. Diese sind in Netzwerk-Komponenten wie Switches und Routern integriert, oder als eigenständige Appliance verfügbar.

Ein einfacher Paketfilter untersucht jedes einzelne IP Paket, das er weiterreichen soll und entscheidet anhand des vom Administrator vorgegebenen Regelwerks, ob das Paket durchge-

lassen wird oder ob es abgewiesen werden muss. Abgewiesene Pakete werden gelöscht, wobei je nach Konfiguration eine ICMP-Nachricht an den Sender verschickt werden kann. In IP basierten Netzen enthält jedes Paket einen IP-Header mit den IP-Adressen der Quelle und des Ziels. Das Regelwerk enthält die IP-Adressen und die verwendeten Ports der erlaubten Kommunikationspartner. Diese einfachen Paket-Filter betrachten also die Header der Ebenen 3 und 4 des ISO-OSI-Modells, um Quelle und Ziel eines Pakets zu bestimmen. Die verwendeten Transportprotokolle können ebenfalls in den Regeln bestimmt werden. Ein Beispiel für ein solches Regelwerk zeigt die folgende Tabelle. Sie regelt den Zugriff auf das interne Netz 192.168.0.0 indem sie nur bestimmte Verbindungen zuläßt. Vorausgesetzt wird, dass durch eine entsprechende allgemeine Regel alle nicht explizit erlaubten Verbindungen unterbunden werden. Erlaubt sind eingehende Mails über den SMTP-Port 25 an den Mailserver 192.168.1.2 (Zeile 1) sowie SSH-Zugriff von extern zu allen Rechner im internen Netz (Zeile 2). Interne Rechner dürfen nach extern nur Verbindungen zum HTTP-Port 80 öffen (Zeile 3) sowie zu einer speziellen Anwendung, welche auf Port 17070 wartet und von Rechnern im Netz 172.16.0.0 angeboten wird.

Zugang	IP Quelle	Port	IP Ziel	Port	Protokoll	Kommentar
erlaubt	*	*	192.168.1.2/16	25	TCP	eingehende Mails
erlaubt	*	*	192.168.0.0/16	22	TCP	ssh von außen
erlaubt	192.168.0.0/16	*	*	80	TCP	ferne Web-Server
erlaubt	192.168.0.0/16	*	172.16.0.0/16	17070	UDP	Anwendung

Tabelle 4.1: Beispiel eines Firewall-Regelwerks

Eine erweiterte Form der einfachen Paket-Filter sind die Stateful-Packet-Filter. Sie betrachten nicht jedes Paket einzeln, sondern können die Pakete einer festen Verbindung zwischen zwei Endpunkten im Netz zuordnen. Wie eine Verbindung definiert ist und wie sie erkannt und behandelt wird, hängt von dem verwendeten Transportprotokoll ab.

Ein Stateful-Packet-Filter kann neben den Port-Nummer auch die Informationen der TCP-Header zum Verbindungsmanagement auswerten und dadurch TCP-Verbindungen erkennen und analysieren. Die Firewall erkennt den Three-Way-Handshake als Verbindungsaufbau und merkt sich diese Verbindung bis zu ihrem Abbau. Es muss daher im Regelwerk auch angegeben werden, in welche Richtung eine erlaubte Verbindung aufgebaut werden darf. Pakete, die zwischen den beiden Endpunkten einer erlaubten Verbindung verschickt werden, können von der Firewall auf sinnvolle Sequenznummern kontrolliert werden, so dass nur zu dieser Verbindung gehörende Pakete akzeptiert werden. Da Pakete schnell einer gespeicherten Verbindung zugeordnet und direkt weitergeleitet werden können, ergibt sich hier ein Geschwindigkeitsvorteil gegenüber der Prüfung jedes Paketes anhand des Regelwerks.

UDP bietet keine Informationen im Header, die eine Verbindung erkennen lassen. Daher ist es nicht wirklich möglich, in einem UDP-basierten Datenaustausch eine Verbindung zu erkennen. Jedes Paket steht für sich und enthält nur IP-Adresse und Port-Nummer vom Sender und vom Empfänger als verwertbare Informationen. Für UDP-Verkehr erstellen Firewalls daher virtuelle Verbindungen. Wie TCP-Verbindungen können diese durch Firewall-Regeln erlaubt oder verboten werden. Eine solche virtuelle Verbindung wird erstellt, wenn ein erstes UDP-Paket weitergeleitet wird. Die Verbindung besteht dann genau zwischen den Endpunkten dieses einen Paketes. Die Endpunkte sind dabei durch zwei Tupel bestehend aus

IP-Adresse und Port-Nummer eindeutig identifiziert. Weitere Pakete, die zwischen denselben Endpunkten verschickt werden, werden als der Verbindung zugehörig betrachtet, solange die Firewall die virtuelle Verbindung in ihrer Verbindungsdatenbank führt. Beendet wird eine solche Verbindung automatisch bei Erreichen eines festgelegten Time-Out-Intervalls, welches durch jedes registrierte Paket wieder neu begonnen wird.

Da TCP- und UDP-Verkehr anders organisiert sind, im Firewall-Regelwerk aber gleich behandelt werden, werden TCP-Verbindungen und virtuelle UDP-Verbindungen im folgenden unter dem Begriff *Session* gemeinsam behandelt. Die Abläufe in einer Firewall sind in Abbildung 4.2 grob dargestellt.

4.1.3 statisches und dynamisches Konfigurieren

Wie bereits erläutert sind zur Durchsetzung der Sicherheitsrichtlinien (vgl. Kapitel 2.1) viele einzelne Regeln nötig. Jede Kombination von einem internen und einem externen Rechner muss mit einer eigenen Regel behandelt werden. Gerade in Grids entsteht so ein großes und kompliziertes Regelwerk. Manche der dort eingesetzten Protokolle, wie zum Beispiel GridFTP, verwenden zudem mehrere Ports gleichzeitig, wodurch im Regelwerk größere Port-Bereiche freigegeben werden müssen. Diese Art der Konfiguration ist nicht nur aufwändig und fehleranfällig, sie kann unmöglich werden, wenn die technischen Grenzen der Firewall erreicht werden. So ist der verfügbare Speicher für das Regelwerk und sonstige Konfigurationen beschränkt. Gängige Limits liegen bei 2 MB. Davon bleiben für das Regelwerk in günstigen Fällen noch 1,5 MB, was in etwa 20000 einzelnen Regeln entspricht. Dies ist ein Wert, der durchaus erreicht werden kann, wenn die Firewall ein größeres Firmennetz vom Internet trennen soll. In der Praxis werden daher oft viele Regeln zu einer einzelnen groben Regel zusammengefasst. Es werden zum Beispiel nicht einzelne Rechner eingetragen, sondern ganze Subnetze und große Port-Bereiche. Dies vereinfacht die Arbeit der Netzwerk-Administratoren erheblich und es verkleinert das Regelwerk, was auch der Performance der Firewall zugute kommt. Es werden aber gleichzeitig unnötige Angriffsflächen und Gefahren geschaffen, denn es sind mehr Rechner und Ports erreichbar als nötig. Je nach Sicherheitsbedürfnis der jeweiligen Organisation, ist das ein inakzeptabler Zustand. Die manuelle Pflege des statischen Regelwerkes durch einen Administrator stellt also keine befriedigende Lösung für Netze dar, deren Rechner Teil eines Grids sind.

Es ist ein Verfahren nötig, das die größten Nachteile dieser Vorgehensweise umgeht. Das bedeutet, dass die Regeln permanent an die sich dynamisch ändernden Bedingungen angepasst werden müssen. Das Regelwerk sollte also dynamisch und immer aktuell sein. Die ständigen Änderungen müssten nicht von einem Administrator gemacht werden, sondern würden automatisiert ausgeführt. Dieses Vorgehen würde die Sicherheit des lokalen Netzes nicht wesentlich gefährden, wenn es durch die Authentifizierung im Grid genau geregelt wird. Zudem könnte die Möglichkeit einer solchen Konfiguration auf bestimmte Rechner beschränkt werden. Wichtig ist dabei, dass genaue Angaben über die Grid-Komponenten gemacht werden, zwischen denen eine Verbindung erlaubt wird. Sonst könnte eine angekündigte Verbindung von einem Dritten missbraucht werden. Beide Seiten der Datenverbindung müssen also eindeutig identifiziert werden. Die Firewall kann dann eine Regel mit zeitlich begrenzter Lebensdauer erstellen, die genau auf die erwartete Verbindung abgestimmt ist. Das in Kapitel 2.3 beschriebene Verfahren stellt eine solche dynamische Lösung dar.

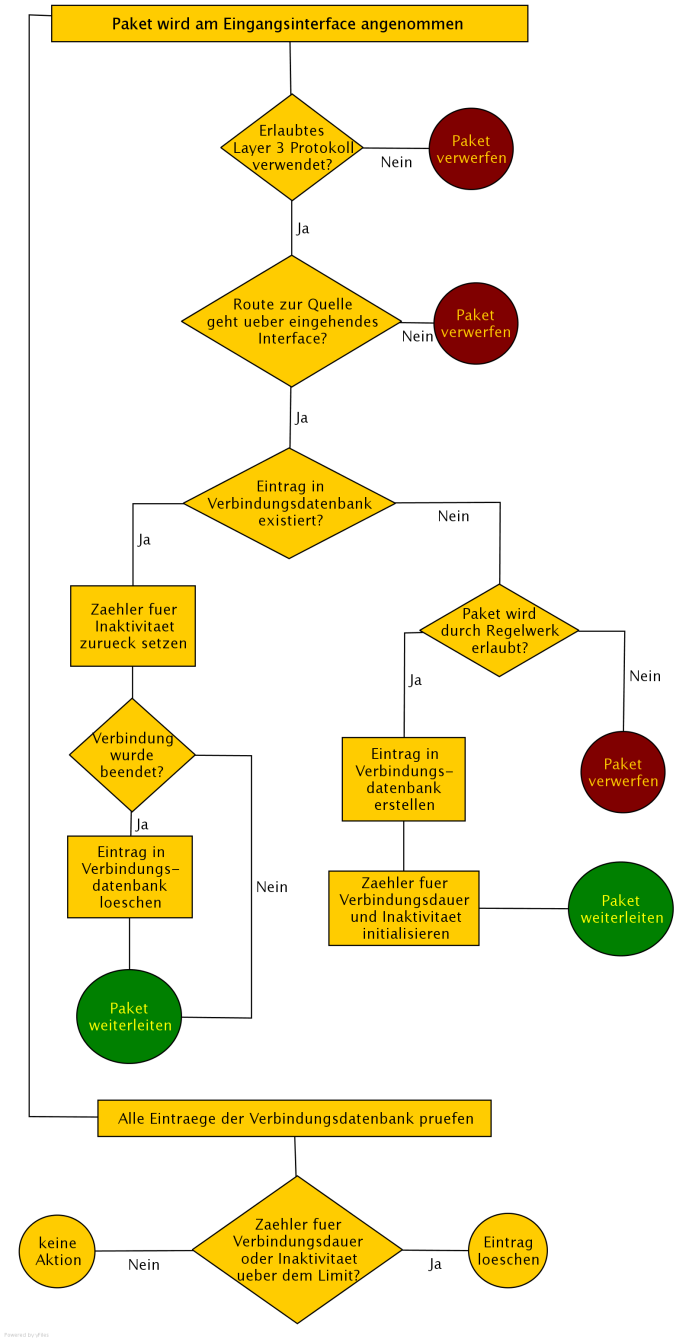

Abbildung 4.2: Funktionsweise einer Firewall

4.2 Parameter einer Session aus Sicht einer Firewall

Eine grundlegende Übersicht zur Funktionsweise von Firewalls wurde in Kapitel 4.1.2 bereits gegeben. Auch ein einfaches Konfigurationsbeispiel war zu sehen. Gängige Firewall-Software kann aber wesentlich genauere und komplexere Regeln befolgen als bisher gezeigt. Die genaue Syntax zur Beschreibung der Regeln unterscheidet sich in den verbreiteten Produkten. Der Funktionsumfang, wie er für die dynamische Verbindungsfreigabe benötigt wird, ist aber ähnlich.

Im Folgenden wird die genaue Konfiguration am Beispiel der Software Netfilter/iptables gezeigt. Das Netfilter Projekt ist für die Paketfilterung im Linux-Kernel zuständig. iptables wird von Administratoren zur Konfiguration des Regelwerks eingesetzt. Durch die weite Verbreitung und die Verfügbarkeit als Open-Source ist iptables für Demonstrationen und Tests gut geeignet. Die hier gezeigten Konfigurationen können auf jedem Standard-PC unter Linux nachvollzogen werden. Vorausgesetzt wird, dass das Netzwerk mit dem Internet Protokoll Version 4 (IPv4 oder einfach IP) als Verbindungsprotokoll arbeitet. Die neuere Version 6 (IPv6) ist noch wenig verbreitet und findet in diesem Beispiel keine besondere Beachtung. Weitere Informationen zu den Neuerungen und Unterschieden finden sich bei [18].

Grundsätzlich teilt iptables den Datenverkehr in drei Kategorien auf. Für jede Kategorie gibt es eine sogenannte Verarbeitungskette (chain), die die zugehörigen Pakete durchlaufen muss. Dies sind die Ketten für eingehende, ausgehende und weitergeleitete Pakete. Für den Datentransfer zwischen Grid-Komponenten ist nur das Weiterleiten interessant. Direkte Verbindungen zur bzw. von der Firewall sind nur für administrative Aufgaben nötig. Für jede Kette kann eine Standard-Aktion festgelegt werden, die auf das Paket angewendet wird, wenn es keiner der konfigurierten Regeln zugeordnet werden kann. Die möglichen Aktionen sind „DROP“ und „ACCEPT“, also Verwerfen oder Weiterleiten. Wenn ein Paket einer Regel zugeordnet werden kann, dann bestimmt die Regel, welche der Aktionen ausgeführt wird. Eine Regel besteht somit aus der Beschreibung der betroffenen Pakete und aus der durchzuführenden Aktion für diese Pakete. Die Konfiguration einer Regel mit den einzelnen Optionen wird im Folgenden erläutert. Dabei wird die Regel schrittweise aufgebaut und Beispiele für das Konfigurationskommando und den resultierenden Eintrag im Regelwerk gegeben.

Vorausgesetzt wird, dass die Standard-Aktion für die „forward-chain“ auf „DROP“ eingestellt ist. Dies ermöglicht eine sichere Konfiguration, in der nur Pakete weitergeleitet werden, die durch eine Regel ausdrücklich erlaubt wurden. Das Kommando zur Erstellung einer neuen Regel lautet

```
iptables -A FORWARD [...]
```

Dabei steht „-A“ für „append“ und „FORWARD“ für die Kette, an die die Regel angefügt werden soll. Diesen grundlegenden Parametern folgen die Optionen zur Charakterisierung der Pakete und der gewünschten Aktion:

Protokoll Das verwendete Transportprotokoll des Paketes kann hiermit festgelegt werden. Mögliche Werte sind TCP, UDP und ICMP, wobei ICMP kein Protokoll zum Transport von Nutzdaten ist und somit hier nicht weiter beachtet wird. Die Option lautet „-p“. Zur Freischaltung einer UDP Session lautet das Kommando:

```
iptables -A FORWARD -p UDP [...]
```

Quelle Diese Option beschreibt den Host, von dem aus das Paket versendet wurde. Er wird anhand seiner IP Adresse oder seines DNS Namens identifiziert. Da sich die IP Adresse eines Grid-Systems während der kurzen Gültigkeit einer dynamischen Regel nicht ändern wird, kann hier die IP Adresse direkt angegeben werden. Die Option lautet „-s“. Beispiel:

```
iptables -A FORWARD -p UDP -s 192.168.13.17 [...]
```

Ziel Genau wie der Sender kann auch der Empfänger mit seiner IP Adresse angegeben werden. Die Option lautet „-d“. Beispiel:

```
iptables -A FORWARD -p UDP -s 192.168.13.17 -d 192.168.123.234 [...]
```

Ports Während die bisherigen Kriterien aus den IP Headern der Pakete gewonnen werden konnten, erlauben Zusatzmodule von netfilter auch die Analyse der Header des Transport-Protokolls. Hier können die erlaubten Port-Nummern angegeben werden. Die Optionen lauten „- - sport“ und „- - dport“ für die Ports bei Sender und Empfänger. Es werden entweder einzelne Ports oder ganze Bereiche angegeben. Das Beispiel erlaubt einen beliebigen Quell-Port oberhalb der Nummer 1024 sowie einen Ziel-Port im Bereich von 10000 bis 11000:

```
iptables -A FORWARD -p UDP -s 192.168.13.17 -d 192.168.123.234
         --sport 1024: --dport 10000:11000 [...]
```

Aktion Es fehlt noch die Angabe über die Aktion, die mit passenden Paketen auszuführen ist. Mögliche Aktionen sind „DROP“ und „ACCEPT“ sowie einige weitere, zum Teil durch Zusatzmodule eingeführte, Aktionen. Da hier eine erlaubte Verbindung beschrieben werden soll, wird „ACCEPT“ gewählt. Die Option lautet „-j“. Das letzte Beispiel ist somit vollständig:

```
iptables -A FORWARD -p UDP -s 192.168.13.17 -d 192.168.123.234
         --sport 1024: --dport 10000:11000 -j ACCEPT
```

iptables kann nicht nur einzelne Pakete untersuchen, sondern auch ganze Sessions verwalten. Dazu wird für ein weitergeleitetes Paket die ensprechende Session in einer Datenbank hinterlegt. Die folgende Anweisung erlaubt alle Pakete, die zu einer gespeicherten Session gehören. Die bisherigen Befehle können somit auch als Charakterisierung der erlaubten Sessions interpretiert werden, mit deren Hilfe die Session bei Kontrolle des ersten Paketes auf Gültigkeit überprüft wird.

```
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Eine Regel kann wieder gelöscht werden, indem derselbe Befehl nochmals abgesetzt wird, mit der Option „-D“ für „delete“ anstelle von „-A“. Der Funktionsumfang von iptables wie auch von anderen Produkten ist wesentlich größer als hier beschrieben. Die gezeigten Optionen stellen die für dynamische Freischaltung von Verbindungen im Grid wesentlichen Punkte dar. Eine umfangreichere Einführung zu diesem Thema bietet [11]. Weitere Optionen sind für die Umsetzung einer Ressource „Verbindungsfreigabe“, wie sie im nächsten Abschnitt beschrieben wird, in eine Firewall-Regel nicht nötig.

4.3 Definition des Dienstes „Verbindungsfreigabe“

Wie in Kapitel 3.4.3 bereits erläutert beinhaltet der WS-Agreement Standard keine domain-specific Erweiterungen für bestimmte Einsatzzwecke. Für den Einsatz zur Reservierung von Verbindungsfreigaben sind also weitere Definitionen nötig. Diese betreffen das XML Format zur Beschreibung der Service Description Terms (SDTs). Darin muss beschrieben werden, zu welcher Leistung sich die Firewall verpflichtet. Die Leistung besteht aus der Zulassung einer bestimmten Session. Zudem ist der Zeitraum der Zulassung festzulegen. Da es sich bei der Freigabe von Verbindungen an einer Firewall um eine sicherheitskritische Funktion handelt ist außerdem zu überprüfen, in wessen Auftrag die Freischaltung erfolgt und inwiefern diese Person berechtigt ist, eine solche Freischaltung zu beantragen.

Basis für die Erweiterungen ist das in Kapitel 3.4.2 besprochene Agreement sowie dessen Factory und das von der Factory bereitgestellte Template. Der Zeitraum der Freischaltung sowie die Session-Parameter werden als SDT beschrieben. Autorisierungsinformationen werden ebenfalls im SDT verpackt. Der Context eines Agreements sieht zwar ebenfalls ein Datum für das Ende der Gültigkeit eines Agreements vor, dieses reicht aber für eine ausreichende Beschreibung nicht aus. Umgesetzt wird die Beschreibung in Form von XML-Schemata, die in der späteren Implementierung durch JAVA Klassen repräsentiert werden.

Freischaltungsdauer

Für die Beschreibung der Zeiten sind zwei verschiedene Abläufe zu betrachten:

- Der Transfer von Daten zwischen zwei Systemen besteht aus drei Aktionen:
  Verbindungsaufbau - Datenaustausch - Verbindungabbau

- Die Freischaltung besteht aus zwei Aktionen:
  Eintragen der Regel - Löschen der Regel

Die Einzelschritte der beiden Abläufe sind voneinander abhängig. Dabei sind zwei verschiedene Gesamtabläufe denkbar. Abbildung 4.3 zeigt diese Abläufe.

Die erste Variante ist offensichtlich. Die Verbindung wird freigeschaltet, anschließend wird die gesamte Kommunikation abgewickelt. Danach wird die Verbindung wieder verboten.

In der zweiten Variante wird die Regel, welche die Verbindung erlaubt, bereits gelöscht, bevor die Kommunikation abgeschloßen ist. Dies ist dann möglich, wenn bereits aufgebaute Verbindungen von der Firewall nicht unterbrochen werden, auch wenn die erlaubende Regel gelöscht wird. Die Verbindungen bleiben in der Verbindungsdatenbank der Firewall eingetragen bis sie regulär beendet werden. Die Regel muss also nur solange existieren, bis die Verbindung aufgebaut wurde.

Welche der beiden Varianten genutzt wird hängt davon ab, wie die konkrete Firewall bei der Löschung von Regeln vorgeht. Die meisten Firewalls verhalten sich wie in der zweiten Variante beschrieben.

Um beide Varianten zu unterstützen, sind bei der Reservierung drei verschiedene Zeitpunkte zu beachten. Dies sind der Zeitpunkt, ab dem die Regel eingetragen ist, der Zeitpunkt an dem die Regel wieder gelöscht wird und der Zeitpunkt, an dem die Kommunikation beendet sein muss. Die letzten beiden Zeitpunkte können, wie oben erläutert, je nach Firewall-Konfiguration zusammenfallen. In diesem Fall ist die Angabe eines Zeitpunktes für das Ende

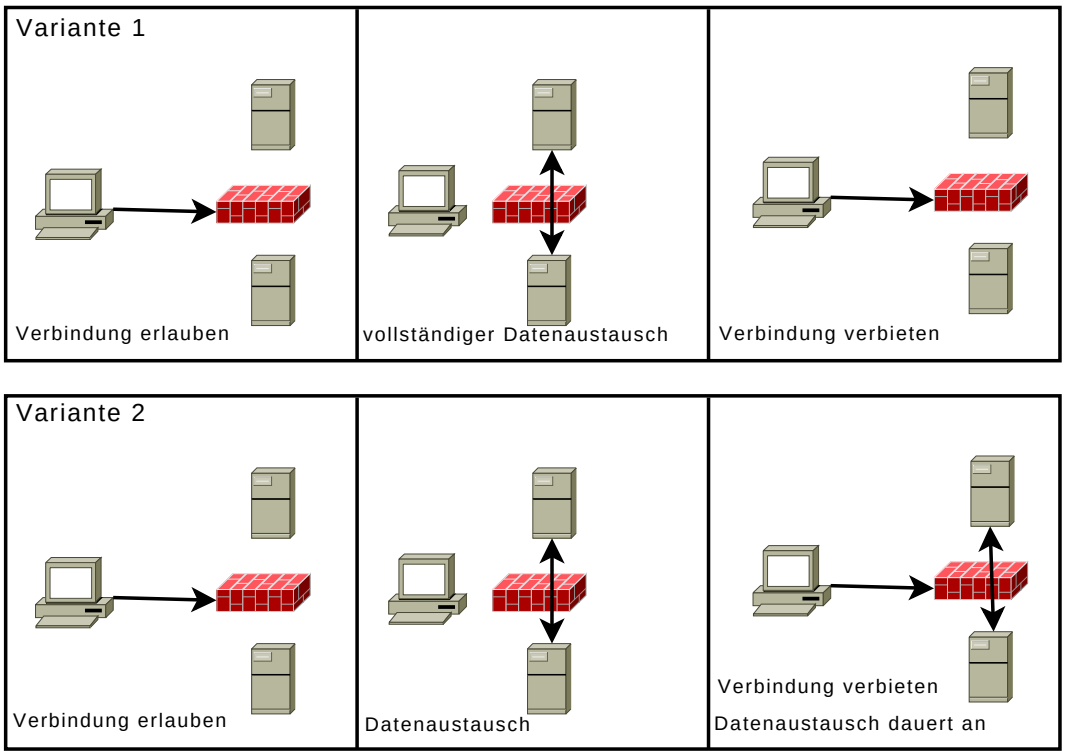

Abbildung 4.3: Abläufe bei einer dynamisch erlaubten Session

der Kommunikation überflüssig. Wenn die Verbindung beliebig lange, also im Extremfall dauerhaft bestehen bleiben soll, dann ist für das Ende ein Zeitpunkt anzugeben, der ausreichend weit in der Zukunft liegt.

Zur Beschreibung der Zeiten wurde ein XML Schema erstellt, das in Anhang A abgedruckt ist. Es beschreibt einen komplexen Datentyp mit drei Elementen vom Typ DateTime, die beliebige Zeitpunkte beschreiben können. Das dritte Element ist optional. Ob die Zeitangaben akzeptabel und sinnvoll sind hat die AgreementFactory zu entscheiden. Tabelle 4.2 stellt den Inhalt des Schemas zusammengefasst dar.

Element	Datentyp	Anzahl	Beschreibung
BeginEstablishment	DateTime	genau 1	Zeitpunkt der Freischaltung
EndEstablishment	DateTime	genau 1	Zeitpunkt der Löschung der Freischaltung
EndConnection	DateTime	0 bis 1	Ende der Kommunikation (optional)

Tabelle 4.2: Aufbau des XML-Datentyps zur Beschreibung wichtiger Zeitpunkte

Sessionbeschreibung

Welche Eigenschaften eine Session aus Sicht der Firewall hat wurde im letzten Unterkapitel anhand der Netfilter Firewall beschrieben. Die Angaben, die dort für die Erstellung einer detailierten Regel notwendig waren müssen als Teil des Agreements beschrieben werden. Es

wurde ein XML Schema für einen Datentyp erstellt, der alle notwendigen Informationen beinhaltet. Das Schema ist in Anhang B abgedruckt. Tabelle 4.3 stellt den Inhalt zusammengefasst dar. Der Inhalt der Aufzählungstypen spiegelt die zur Zeit üblichen Protokolle wieder. Eine spätere Erweiterung für neue Protokolle ist leicht möglich.

Da im Grid-Umfeld oft Verfahren benutzt werden, die Daten über mehrere parallele Verbindungen übertragen, ist die Möglichkeit enthalten, für eine Session mehrere zusammendhängende Ports zu öffnen. Einzelne Ports werden angegeben, indem Anfang und Ende des Port-Bereiches den gleichen Wert haben. Nicht zusammenhängende Ports müssen durch mehrere Sessions beschrieben werden. Ob die Adressen und Port-Nummern akzeptabel und sinnvoll sind, hat die AgreementFactory zu entscheiden.

Element	Datentyp	Beschreibung
NetworkProtocol	ConnectionNetworkProtocolType	Das Verbindungsprotokoll (IPv4 oder IPv6)
TransportProtocol	ConnectionTransportProtocolType	Das Transportprotokoll (TCP oder UDP)
SourceAddress	ConnectionAddress Type	Host-Adresse als String
DestinationAddress	ConnectionAddressType	Host-Adresse als String
SourcePortRangeStart	ConnectionPortNumber Type	Begin des Portbereiches beim Client
SourcePortRangeEnd	ConnectionPortNumberType	Ende des Portbereiches beim Client
DestinationPortRangeStart	ConnectionPortNumberType	Begin des Portbereiches beim Server
DestinationPortRangeEnd	ConnectionPortNumberType	Ende des Portbereiches beim Server

Tabelle 4.3: Aufbau des XML-Datentyps zur Beschreibung der Session

Informationen zur Autorisierung

Mit den bisherigen Informationen sind Zeitpunkt und Umfang der Freischaltung ausreichend beschrieben. Es werden aber keine Informationen über den Nutzer bereit gestellt, wozu er die Freischaltung benötigt und was ihn zur Beantragung der Freischaltung berechtigt. Zur technischen Umsetzung der Freischaltung sind diese Informationen auch nicht nötig. Da es sich hier aber um eine Funktion handelt, die für das jeweilige Unternehmen eine hohe sicherheitstechnische Bedeutung hat, ist eine umfangreiche Autorisierung der Nutzer unbedingt nötig.

Zur Authentifizierung und Autorisierung des Nutzers bringt Grid Middleware meistens eingebaute Mechanismen mit. Zertifikate sind eine übliche Technik hierfür. Im Falle einer Firewall-Konfiguration reicht es aber möglicherweise nicht aus, sicherzustellen, dass die betreffende Person ein legaler Benutzer des Grids ist. Insbesondere, wenn die virtuelle Organisation aus Mitarbeitern mehrerer Unternehmen an verschiedenen Standorten besteht ist eine detaillierte Autorisierung aufgrund diverser Kriterien nötig. Diese muss flexibel sein, um auf die verschiedensten Gegebenheiten reagieren zu können. Zur Freischaltung gehören also

neben ihrem Zeitpunkt und Umfang auch Informationen über den Antragsteller und dessen Berechtigung.

Eine standardisierte und weit verbreitete Möglichkeit zur Beschreibung der nötigen Informationen ist die *Security Assertion Markup Language* (SAML) [7]. Sie ist ein von der *Organization for the Advancement of Structured Information Standards* (OASIS) entwickelter Standard und basiert auf XML. Der Standard besteht aus drei Teilen: Den eigentlichen Sicherheitsinformationen (*assertions* genannt), Anfrage-/Antwort-Protokollen zum Austausch der Daten und sogenannten Bindings, die beschreiben, wie die Nachrichten über diverse Protokolle der Anwendungsschicht wie HTTP oder SOAP transportiert werden. Die einzelnen Schichten werden in einem Profil zusammengefaßt und beschrieben. Da der Transport der Informationen hier bereits geregelt ist, besteht dieses Profil aus den von der Middleware eingesetzten Technologien wie HTTP, SOAP, WS-Agreement und den SDTs. Neu dazu kommen lediglich die eigentlichen *assertions*.

Die Assertions enthalten neben den Identifikationsmerkmalen der authentifizierten Person noch zwei weitere Arten von Informationen. Dies sind Attribute und Autorisierungsentscheidungen. Attribute können beliebige Informationen zum Benutzer sein, wie z.B. Zugehörigkeit zu einer bestimmten realen Organisation oder Arbeitsgruppe. Autorisierungsentscheidungen können z.B. die Erlaubnis zum Zugriff auf ein bestimmtes System enthalten. Wie diese Aussagen strukturiert sind und welche Informationen sie genau enthalten hängt von der verwendeten Grid-Middleware und den Strukturen in der VO ab.

Durch die XML-Basis kann eine solche *assertion* einfach als dritte Komponente in den SDT eingebunden werden. Der Abgleich der Informationen mit den Sicherheitsrichtlinien kann z.B. über die im SAML Standard vorgesehenen Metadaten geschehen, welche entsprechende Konfigurationsdaten enthalten können. Die Factory übernimmt diese Aufgabe.

Verwaltet und zur Verfügung gestellt werden SAML-Informationen in einer VO durch einen besonderen Service, den *Identity Provider*.

Service Description Terms

Ein Service Description Term (SDT) zur Beschreibung einer Freischaltung beinhaltet in seinem beschreibenden Teil somit drei Elemente komplexer Datentypen. Dies sind ein Element zur Beschreibung der Freischaltungsdauer, ein Element zur Beschreibung der Session und (optional) als drittes Element die beschriebene SAML Komponente (in dieser Reihenfolge). Ein Agreement kann mehrere Sessions umfassen, indem es mehrere SDTs enthält.

4.3.1 Verlauf einer Freigabe

Zusammenfassend soll hier der Ablauf einer Verbindungsfreigabe in Form eines Agreements nochmals dargestellt werden. Die Komponenten und vielfältigen Möglichkeiten zur Reservierung in WS-Agreement wurden bereits in Kapitel 3.4.2 vorgestellt. Abbildung 4.4 zeigt, wie der Ablauf bei der einfachen Reservierung von Verbindungsfreigaben konkret aussieht.

Der GridScheduler in seiner Funktion als Initiator fordert von der AgreementFactory des Firewall-Agenten ein Template an. Dieser liefert ein entsprechendes Template. Der Scheduler erstellt nun einen Vorschlag für ein Agreement und sendet dieses als AgreementOffer an die Factory. Diese lehnt den Vorschlag entweder ab (hier nicht dargestellt), woraufhin der Scheduler ein neues Angebot sendet, oder sie nimmt den Vorschlag an. Aus einem akzeptablen Vorschlag erstellt die Factory ein Agreement und sendet dem Scheduler eine *Endpoint Refe-*

rence (EPR) auf das Agreement-Objekt. Die technische Umsetzung der Konfiguration wird gleichzeitig mit der Erstellung des Agreements von der Firewall durchgeführt. Ein Agreement hat eine festgelegte Lebensdauer. Ist diese abgelaufen, dann wird das Agreement-Objekt von der Engine zerstört und dabei dessen Methode zur Terminierung der Freischaltung aufgerufen. Das Agreement-Objekt entfernt die Konfiguration dann aus der Firewall. Gleiches passiert, wenn die Methode zur Terminierung vorher explizit aufgerufen wird, was beiden Parteien offen steht, um das Agreement vorzeitig aufzukündigen. Änderungen und Erweiterungen von bestehenden Agreements sind aus Gründen der Sicherheit und einfachen Nachvollziebarkeit nicht vorgesehen. Nachverhandlungen sind daher nur indirekt durch Vereinbahrung eines neuen Agreements und Terminierung des bisherigen möglich. Viele Firewalls bieten auch die Möglichkeit, einer Konfiguration eine zeitliche Beschränkung mitzugeben, so das eine manuelle Terminierung nur bei vorzeitiger Kündigung nötig ist. Diese Funktion der Firewall verhindert auch, dass Freigaben auf Dauer erhalten bleiben, wenn der Agent plötzlich nicht mehr verfügbar ist, z.B. wegen Systemabstürzen. Steht diese Funktion nicht zur Verfügung, dann muss nach einem technischen Problem anhand der Logging-Informationen überprüft werden, welche Freigaben zu entfernen sind. Ein Absturz des Agenten kommt dann einem vorzeitigen Aufkündigen aller Agreements gleich.

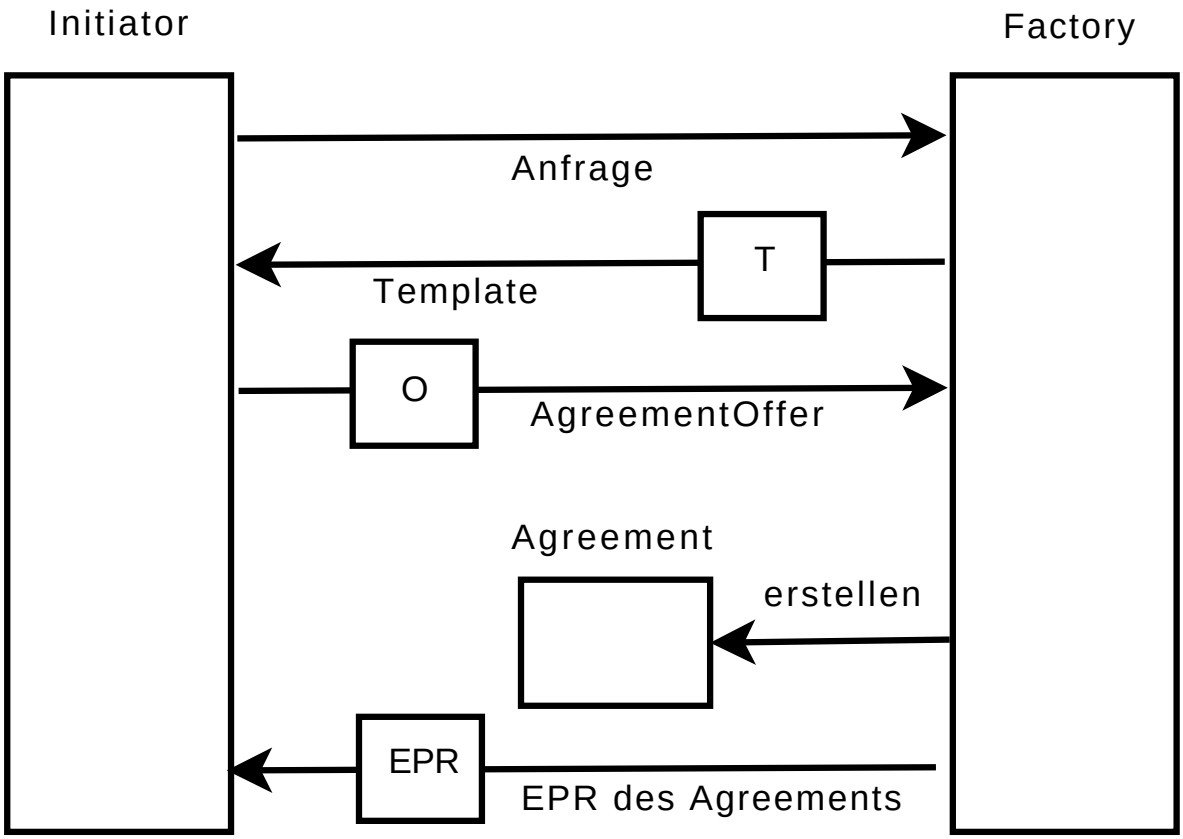

Abbildung 4.4: Ablauf einer einfachen Reservierung mittels WS-Agreement

Beispiel

Abbildung 4.5 zeigt ein konkretes Agreement als Beispiel. Die Reservierung durchläuft vier Stationen, die hier einzeln dargestellt sind. Der Inhalt des Kontext ist nicht angegeben, da er keine für das Verfahren spezifischen Informationen enthält. Die Reservierung beginnt mit der Abfrage der Templates durch den Scheduler. Die entsprechende Anfrage enthält noch keine spezifischen Informationen. Anschließend schickt die Factory ein Template. Dieses beinhaltet Default-Werte, die bereits den Rahmen der nötigen und akzeptierten Werte beschreiben. Im gezeigten Fall werden Verbindungen von beliebiger Stelle in das eigene Netz (hier beispielhaft das eigentlich private Netz 10.10.0.0) akzeptiert, bei beliebiger Port-Wahl. Der Scheduler erstellt nun ein Angebot für ein Agreement. Dieses beinhaltet die drei beschriebenen domain specific Komponenten. Das sind die Sessionbeschreibung mit IP-Adressen und Port-Nummern, die Freischaltungsdauer, bestehend aus zwei Zeitpunkten, und die SAML Informationen des beantragenden Nutzers, hier bestehend aus Name und zugehöriger Arbeitsgruppe. Wenn die Factory das Angebot an nimmt, dann erstellt sie daraus ein Agreement mit gleichem Inhalt.

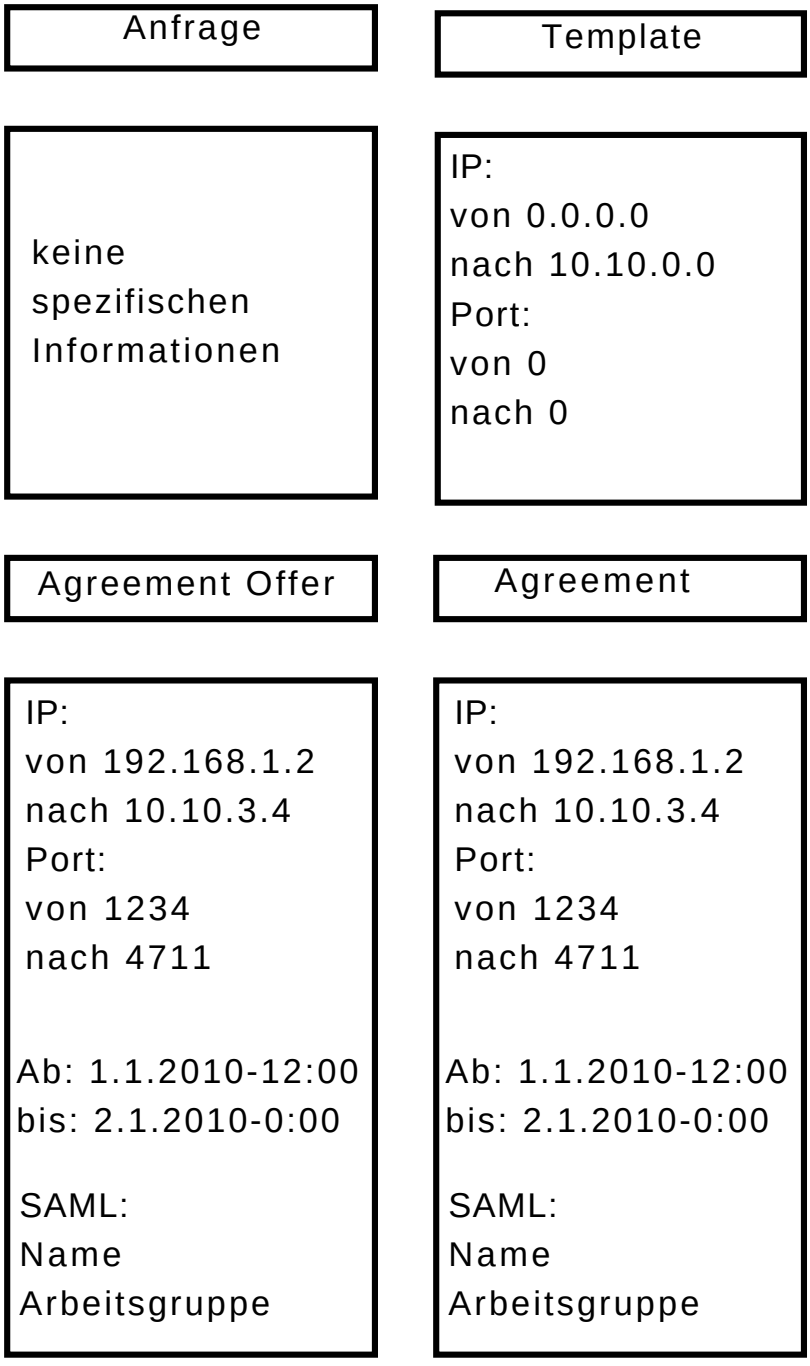

Abbildung 4.5: Zustände des Agreements während der Reservierung

Kapitel 5

Implementierungen

In diesem Kapitel wird auf die Implementierung der vorgestellten Techniken auf Basis von WS-Agreement eingegangen. Dabei wird keine vollständige Software zur Verhandlung und Freischaltung von Verbindungen mit anschließendem Transfer implementiert. Eine solche Implementierung muss immer Teil einer konkreten Grid-Middleware sein und kann nicht für sich alleine stehen. Die Middleware muss einen Grid-Scheduler enthalten der mittels WS-Agreement Reservierungen vornimmt. Zur Zeit ist aber keine praxistaugliche Grid-Software verfügbar, die einen geeigneten Rahmen bietet. Die Projekte zur Implementierung und Integration des Standards befinden sich noch in der Entwicklung. Anstatt einer Implementierung innerhalb einer bestimmten Grid-Middleware wird daher eine allgemein nutzbare Variante vorgestellt, die in jede Grid-Middleware integriert werden kann. Dazu wird eine allgemeine Implementierung des WS-Agreement Standards, die wsag4j Bibliothek, um die für das vorgestellte Verfahren spezifische Funktionalität erweitert. Die Ergebnisse können in Projekten zur Integration von MetaScheduling-Funktionalitäten für Netzwerk-Ressourcen in Grid-Middleware genutzt werden. Die Klassen sind auch unabhängig von der verwendeten Strategie zur dynamischen Verhandlung über Ressourcen zu sehen.

5.1 Die wsag4j Implementierung

WS-Agreement ist eine Spezifikation. Sie beinhaltet keine einsatzfähige Implementierung der beschriebenen Konzepte. Eine Implementierung wird zur Zeit im wsag4j-Projekt entwickelt [14]. wsag4j bietet ein Framework für die Entwicklung von Diensten zur Reservierung von Ressourcen. Dazu implementiert es die wesentlichen Bestandteile des Standards in der Sprache Java.

Kern der Software ist die Engine, welche die grundlegenden Funktionalitäten bereitstellt. Sie nutzt weitere WS-* Implementierungen wie z.B. WS-Security zur Signierung der ausgetauschten Nachrichten. Es können mehrere Engines auf einer Maschine laufen. Sie sind hauptsächlich für die Verwaltung von Informationen und Konfigurationen zu ihren *AgreementFactories* und zu den laufenden *Agreements* zuständig. Zudem bieten sie zusätzliche Funktionalitäten wie z.B. die automatisierte Kontrolle von SDTs anhand von *AgreementCreationConstrains*.

Eine Engine kann beliebig viele AgreementFactories betreiben. Diese können beliebig viele Agreements erstellen. Das Framework stellt dazu Interfaces für die *AgreementFactory-*, *Agreement-* und *AgreementState-* PortTypes bereit. Neben den Interfaces existieren rudimentäre Implementierungen von Klassen als Beispiel und Hilfe für Entwickler. Die eigentliche Funktionalität, also Vereinbaren und Umsetzen eines konkreten *Agreements*, ist aber vom Anwendungsfeld abhängig und daher nicht umgesetzt. Diese muss hier in Form von domain-specific Erweiterungen implementiert werden. Die Grundfunktionen der Engine und die bereitsgestellten Interfaces bilden dafür eine intuitiv und ohne detaillierte Kenntnisse der internen Abläufe nutzbare Basis.

Die folgenden Klassen und Strukturen müssen neu implementiert werden:

Agreement Die Klasse *ConnectionAgreement* soll das konkrete *Agreement* darstellen und enthält die für den Anwendungsfall spezifischen Informationen.

Factory Die Klasse *ConnectionAgreementFactory* wird eine für den Anwendungsfalls spezifische *Factory*, die entsprechende Templates bereit stellt und *Agreements* vom Typ *ConnectionAgreement* erstellen kann.

Datenstrukturen Die für den Anwendungsfall spezifischen Informationen im *Agreement* müssen durch entsprechende Klassen repräsentiert werden.

Funktion Die Hauptfunktionalität, also die Konfiguration der vereinbarten Freischaltung, ist in der *Factory* zu implementieren.

Reservierungen für Verbindungsfreigaben sind unkompliziert. Die erforderliche Implementierung ist kurz und einfach gehalten. Das folgende Unterkapitel beschreibt die neuen Klassen und ihre Funktionalität.

5.2 Eine allgemeine Implementierung

Datenstrukturen

Die Datenstrukturen zur Verwaltung der für den Anwendungsfall spezifischen Funktionen liegen bisher nur als XML-Schema vor. Zur besseren Behandlung der XML Dokumente im Java Code müssen Java Klassen generiert werden, die die XML Daten repräsentieren. Das XML-Beans Projekt bietet Werkzeuge zur automatischen Generierung dieser Klassen auf Basis der Schema Dateien. Mir diesem Hilfsmittel wird für jeden der neu definierten XML Datentypen eine Klasse erzeugt. Diese Java Klassen werden in den im Folgenden beschriebenen Klassen zur Verarbeitung der auszutauschenden XML Daten genutzt.

Klassen

Die spezifische Implementierung besteht aus den beiden Klassen „ConnectionAgreement“ und „ConnectionAgreementFactory“, welche eine konkrete Implementierung der vorgegebenen Interfaces darstellen. Die Factory hat die Aufgaben, auf Anfrage ein Template bereit zu stellen, Agreement Angebote zu prüfen und bei positiver Prüfung ein Agreement zu erstellen. Die Factory erstellt bei Konstruktion des Agreement eine Firewall-Regel und fügt diese in das Firewall-Regelwerk ein. Läuft das Agreement ab, oder wird es vorzeitig beendet, dann entfernt es die Regel aus dem Firewall-Regelwerk. Die Abbildungen 5.1 und 5.2 zeigen die Klassendiagramme der beiden Klassen.

AgreementFactory.getTemplates()

Die Methode *getTemplates()* der AgreementFactory liefert bei Aufruf ein AgreementTemplate für eine Freischaltung zurück. Der Inhalt des Template, also vor allem die Default-Werte werden in der aktuellen Version direkt im Quellcode gesetzt. Später könnte dies über Konfigurationsdateien erfolgen, um den Quellcode bei verschiedenen Einsatzszenarien nicht ändern zu müssen. Da die Methode nicht auf Parameter reagieren muss und das Ergebnis immer identisch ist, ist ihr Code sehr einfach gehalten. Das Struktogramm in Abbildung 5.3 zeigt die Abläufe.

AgreementFactory.createAgreement()

Die Methode *createAgreement()* erhält als Parameter ein AgreementOffer, also ein fertiges Agreement als Vorschlag. Dieser ist zu überprüfen und entweder in ein richtiges Agreement zu überführen oder abzulehnen. Die Ablehnung erfolgt durch eine Fehlermeldung, genauer

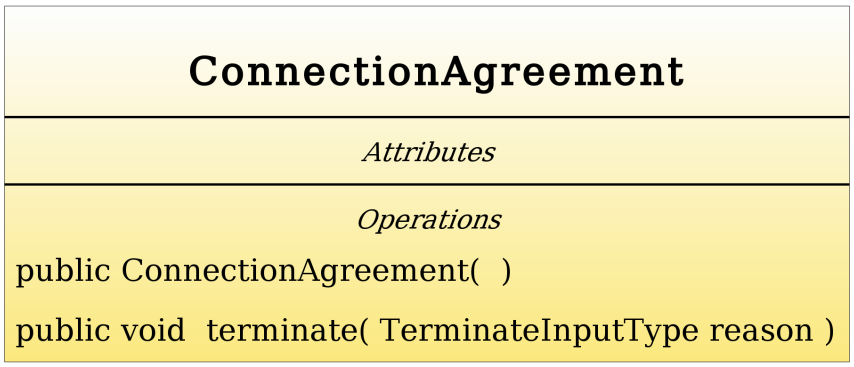

Abbildung 5.1: Klassendiagramm der neuen Agreement Klasse

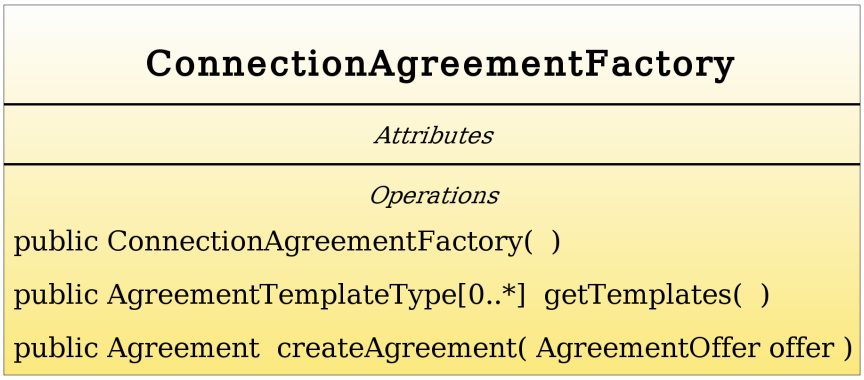

Abbildung 5.2: Klassendiagramm der neuen Agreement Factory

gesagt durch das Auslösen einer Exception. Das Struktogramm in Abbildung 5.4 zeigt die Abläufe in der Methode.

Das AgreementOffer wird erst auf die Existenz von SDTs überprüft. Anschließend werden die SDTs, es können einer oder mehrere vorhanden sein, überprüft. Sind die enthaltenen Parameter akzeptabel, dann werden sie in ein Agreement Objekt überführt. Dieses wird als Ergebnis zurück gegeben. Zuvor wird die eigentliche Freischaltung vorgenommen, indem die Firewall entsprechend konfiguriert wird. Diese Konfiguration ist vom Typ der Firewall abhängig und daher in dieser allgemeinen Implementierung nicht ausgeführt. Eine beispielhafte Umsetzung für *iptables* findet später in diesem Kapitel statt.

Die SDTs können auf korrekte Syntax und Wertebereiche überprüft werden. Zudem sind weitere, vom Einsatzszenario abhängige Kriterien für die Akzeptanz der Freischaltungen möglich. Solche Kriterien, z.B. genauere Regeln für die Autorisierung mittels der eingetragenen Assertions, sind in dieser Version noch nicht implementiert. Später könnten sie in Konfigurationsdateien beschrieben werden, um den Quellcode nicht anpassen zu müssen. Zudem

AgreementFactory - getTemplates()

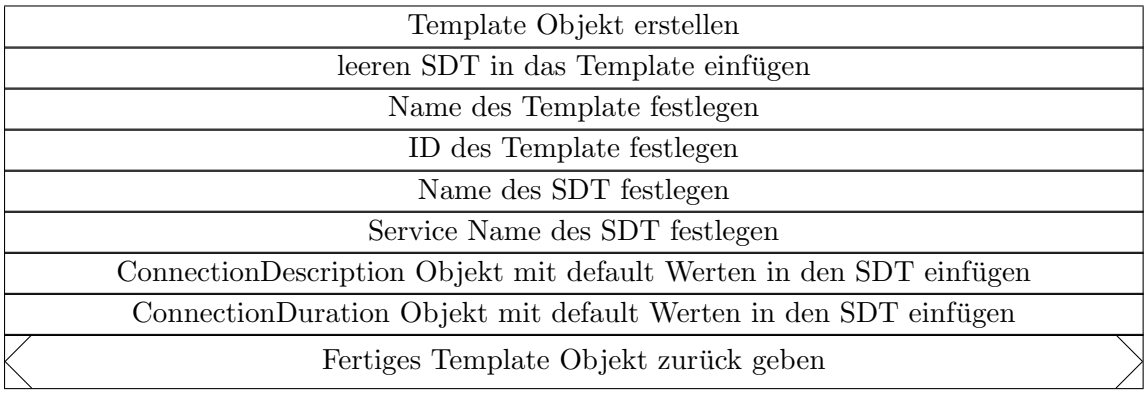

Abbildung 5.3: Struktogramm zur Methode *AgreementFactory.getTemplates()*

AgreementFactory - createAgreement()

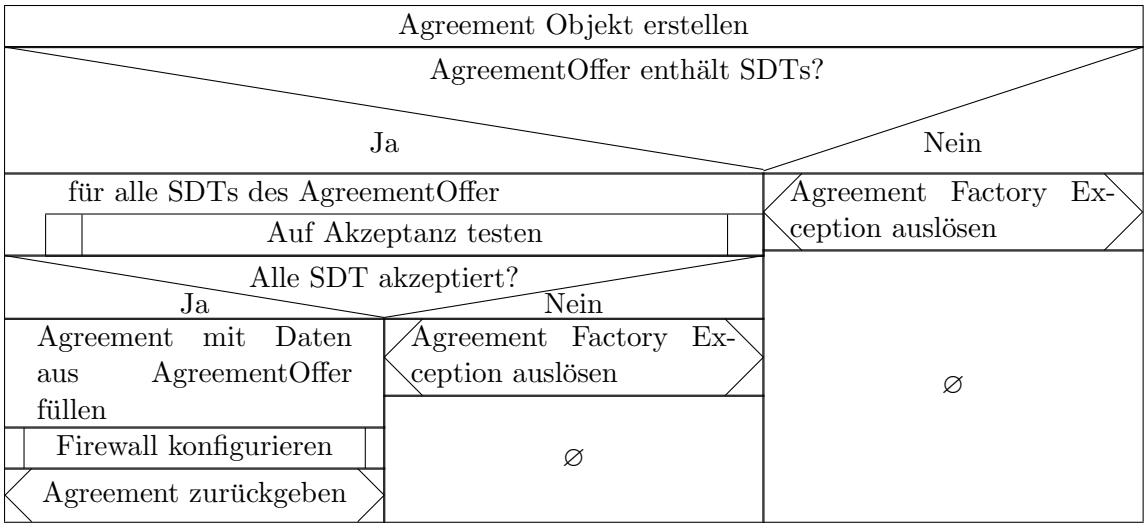

Abbildung 5.4: Struktogramm zur Methode *AgreementFactory.createAgreement()*

können die Möglichkeiten der wsag4j-Engine zur Evaluierung genutzt werden.

Eine Archivierung der Inhalte des Agreements durch die Factory ist hier sinnvoll, um spätere Fehlersuche und Sicherheitsüberprüfungen zu erleichtern.

ConnectionAgreement.terminate()

Die Methode *terminate()* entfernt die Freischaltung wieder aus der Firewall. Dies geschieht, wenn das Agreement vorzeitig beendet wird, oder wenn das Agreement regulär ausläuft. Hierzu wird aus den SDTs ein entsprechendes Kommando generiert und an die Firewall gesendet. Dieses vorgehen ist Firewall-spezifisch. Eine beispielhafte Implementierung für *iptables* wird im Anschluss besprochen.

Beispielmethode: iptablesCommand()

Als Beispiel für die Konfiguration einer Firewall auf Basis eines ConnectionAgreement soll die Funktion *iptablesCommand()* dienen. Sie erzeugt aus den Informationen eines *ConnectionDescriptionType* Objektes eine Befehlszeile für Unix-Shells. Dabei ist über einen zweiten Parameter wählbar, ob das erzeugte Kommando die Regel anlegen oder entfernen soll. Dieses Kommando muss anschließend von der aufrufenden Methode auf einer Unix-Shell ausgeführt werden.

Die fertige Befehlszeile wird per ssh eine Verbindung zur Firewall herstellen und mittels Eingabeumleitung den *iptables* Befehl inklusive Parameter absetzen. Anschließend wird die ssh Verbindung geschlossen. Wie ein *iptables* Kommando aussieht wurde bereits in 4.2 gezeigt. Die folgende Tabelle zeigt den Inhalt eines möglichen *ConnectionDescriptionType* Objektes.

NetworkProtocol	TransportProtocol	SourceAddress	DestinationAddress
IPv4	TCP	192.168.13.17	192.168.123.234
SourcePortStart	SourcePortEnd	DestinationPortStart	DestinationPortEnd
4711	4711	1707	1707

Tabelle 5.1: Beispiel für eine Session

Aus dieser Tabelle erzeugt die Funktion das folgende Kommando zum Anlegen der Regel:

```
echo "iptables -A FORWARD
             -p TCP
             -s 192.168.13.17
             -d 192.168.123.234
             --sport 4711:4711
             --dport 1707:1707
             -j ACCEPT ; exit" |
ssh firewall.my-domain.de
```

Der Text „firewall.my-domain.de“ ist im Quellcode fest einprogrammiert, könnte später aber auch über Konfigurationsdateien eingelesen werden. Diese Methode ist die einzige Komponente im System, die eine Firewall-Konfiguration vornimmt. Sie muss die entsprechenden administrativen Rechte haben, um auf die Firewall zugreifen zu können. Die Anmeldung beim

ssh-Server der Firewall kann nicht mit einem manuell angegebenen Passwort geschehen. Das Hinterlegen des Passworts im Klartext ist zu unsicher. Daher werden Schlüssel-Paare zur gegenseitigen Identifikation genutzt. Diese erlauben sicheres und automatisches Verbinden. Das Struktogramm in Abbildung 5.5 zeigt die Abläufe in der Funktion.

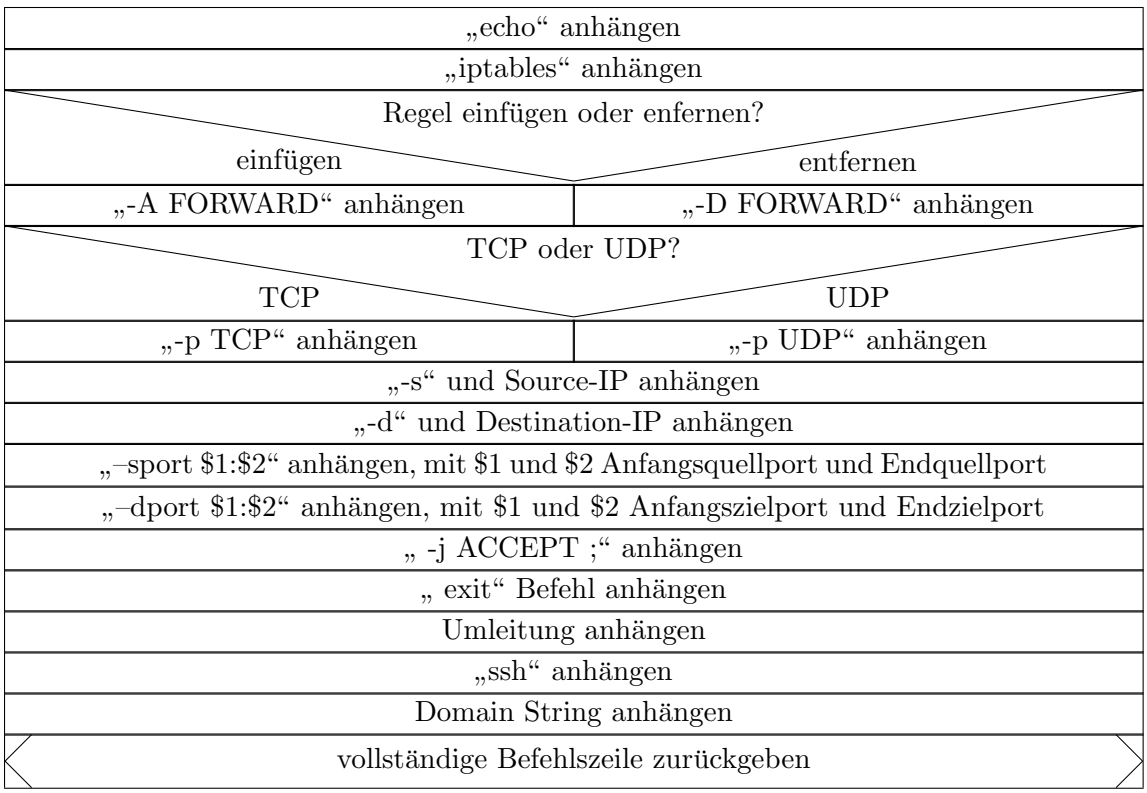

Abbildung 5.5: Struktogramm zur Funktion *iptablesCommand()*

Wie bereits erwähnt muss die beschriebene Implementierung im praktischen Einsatz angepasst werden. Die Kriterien zur Akzeptanz der Freischaltung müssen beschrieben werden und das Erstellen und Ausführen des Konfigurationskommandos ist an die verwendete Hardware und Software anzupassen. Die Abläufe sind aber allgemein gehalten und universell anwendbar. Weitere allgemeine Überlegungen zum praktischen Einsatz folgen in Kapitel 7.

Die neue Implementierung umfasst mit der Factory, dem Template, dem Agreement, der Prüfung auf Akzeptanz, der Freischaltung und der Beendigung der Freischaltung alle nötigen Vorgänge für den praktischen Einsatz. Alle nötigen Datenstrukturen werden zur Verfügung gestellt.

Kapitel 6

Architektur einer Software zur dynamischen Verbindungsfreigabe

Dieses Kapitel beschreibt die Archtitektur und die Funktionsweise eines Systems zur dynamischen Freischaltung von Transport-Verbindungen als Grid-Ressource mittels eines Meta-Scheduling Systems. Dabei wird zuerst ein einfaches Szenario behandelt. Anschließend wird das Konzept im Umfeld einer weitreichenden und umfassenden Lösung zur dynamischen Nutzung von allgemeinen Netzwerkressourcen betrachtet und in dessen Architektur eingegliedert.

6.1 Einfacher Ansatz

Ein einfaches Szenario für den Einsatz dynamischer Verbindungsfreischaltung zeigt Abbildung 6.1. Anhand dieses Szenarios sollen die Komponenten und Abläufe allgemein beschrieben werden. Das Grid ist hier wie üblich in Sites aufgeteilt, also in mehrere Standorte. An jedem Standort befinden sich mehrere Grid-Komponenten. Die lokalen Netze an den Sites sind nach außen hin mit mindestens einer Firewall gesichert. Mehrere Firewalls pro Netz sind bei mehreren Anbindungen an externe Netze nötig, oder bei einer weiteren Unterteilung des Netzes in Teilnetze. In jedem Fall untersteht der gesamte Standort aber einer einzigen Organisation.

Verbunden sind die Sites durch ihre Anbindung an das INTERNET. Dieses wird als undurchsichtige Wolke betrachtet, welche eine uneingeschränkte Konnektivität zwischen den Sites gewährleistet. In der Praxis könnten hier anstatt des INTERNET auch spezielle Netze mit besonderen Funktionaltäten aber auch besonderen Anforderungen an die Konfiguration genutzt werden. Dieses sehr komplexe Szenario wird erst im nächsten Unterkapitel betrachtet. Bis auf weiteres erfordert das verwendete externe Transit-Netz keine zu beachtende Konfiguration.

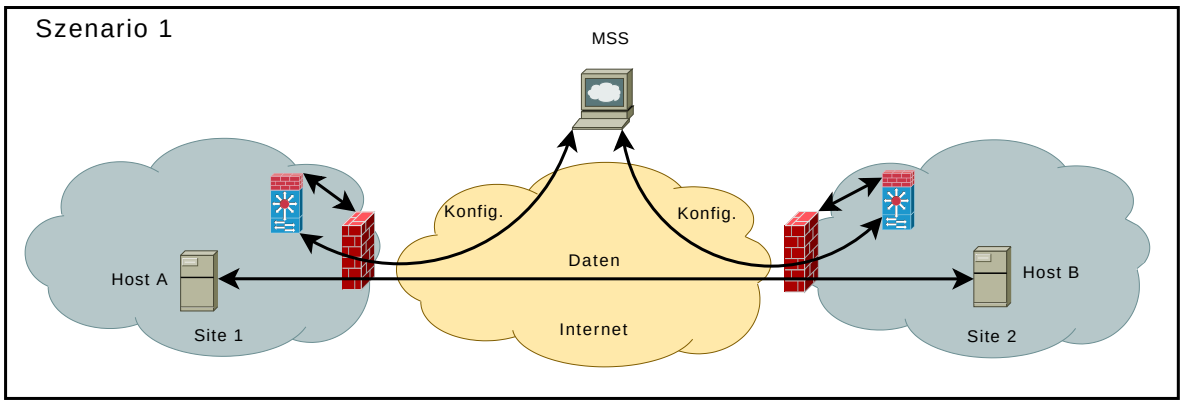

Abbildung 6.1: Szenario 1

6.1.1 Komponenten

Der Client

Der Client erstellt in einem Grid die Aufgaben und reicht sie an die Systeme weiter, auf denen sie dann als Jobs abgearbeitet werden. Dazu können mehrere Teil-Jobs zu Workflows zusammengefasst werden, die sich über verschiedene Systeme verteilen. Neben der eigentlichen Berechnung enthalten solche Workflows auch den Transfer von Dateien zwischen den einzelnen

Systemen. Die Reservierung der nötigen Ressourcen durch den Client erfolgt über einen Grid-weiten Scheduler, der im folgenden Abschnitt betrachtet wird.

Der GridScheduler

Der GridScheduler oder Meta-Scheduling-Service (MSS) ist die zweite wichtige Komponente der Architektur. Er stellt die Funktionen bereits, die in Kapitel 3.3 besprochen wurden. Er reserviert also bei den Systemen im Grid diverse Ressourcen wie CPU-Zeit, Arbeitsspeicher und Massenspeicher. Initiiert wird eine solche Reservierung durch die Anfrage eines Clients. Dieser erstellt einen Workflow, für den die verschiedenen Ressourcen zu bestimmten Zeiten in einer festgelegten Reihenfolge zur Verfügung stehen müssen. Neben den System-Ressourcen gibt es mehrere Typen von Netzwerk-Ressourcen, darunter die Bandbreite und die Verbindungsfreigabe. Beide werden für den Transfer von Dateien benötigt. Das vorgestellte Konzept erweitert die Ressourcen, die beim MSS reserviert werden können, um die Verbindungsfreigabe.

Wenn ein Dateitransfer zwischen zwei Systemen A und B stattfinden soll, dann hat der MSS eine entsprechende Verbindungsfreigabe bei den Firewalls zu beantragen, die auf dem Weg zwischen A und B liegen. Im Modell sind das die Firewalls der beiden Sites, in denen sich diese Systeme befinden. Sollten beide Systeme zu derselben Site gehören, dann ist gar keine Reservierung nötig. Der MSS spricht also mit den Firewall-Agenten der Sites, die im nächsten Abschnitt vorgestellt werden. Dazu nutzt er die Mechanismen des WS-Agreement Standards. Der MSS ist somit der Client bzw. Initiator eines Agreements, das er über die AgreementFactory des Agenten erstellen muss.

Der Firewall-Agent

Ein wesentlicher Bestandteil der Architektur ist der Agent, der die Firewall konfiguriert. Er hat dazu direkten Kontakt zur Firewall. Diese Komponente kann Teil der Firewall-Hardware sein, oder eine eigenständige Hardware-Komponente. Erste Implementierungen werden aber vermutlich ohne Beteiligung von Firewall-Herstellern durchgeführt, so dass die Betrachtung einer eigenständigen Komponente realistischer ist. Diese befindet sich im internen Netz und kommuniziert mit der Firewall, wobei sie ein Firewall-spezifisches Protokoll nutzt. Alle handelsüblichen Firewalls bieten dazu eine proprietäre Schnittstelle, die zum Beispiel Konfigurationskommandos über eine ssh-Verbindung entgegennehmen kann. Hierüber teilt der Agent der Firewall die konkreten Verbindungsfreigaben mit und widerruft sie auch bei Bedarf wieder. Dafür wird nur die Standard-Funktionaliät genutzt. Eine Anpassung oder Erweiterung der Firewall ist nicht nötig. Im letzten Kapitel wurden Formulierung und Ausführung eines solchen Kommandos am Beispiel von iptables bereits beschrieben. Der Agent ist somit die einzige Komponente, die direkten Kontakt zur Firewall hat. Alle anderen Grid-Komponenten müssen sich an den Agenten wenden.

Es kommt vor, dass in einem Netz mehrere Firewalls betrieben werden, z.B. weil es mehrere Anbindungen an externe Netze gibt. Da diese Firewalls alle unter einer Kontrolle stehen reicht ein einziger Agent pro Site aus, der dann alle Firewalls konfigurieren kann.

6.1.2 Konfiguration

Der neue Firewall-Agent sowie die erweiterten Funktionen des MSS müssen bei Betrieb und Konfiguration des Grid beachtet werden. Es sind weitere Informationen, wie Vorhandensein und Zuständigkeit der Firewall-Agenten, im Grid bereit zustellen. Heutige Grid-Software hält

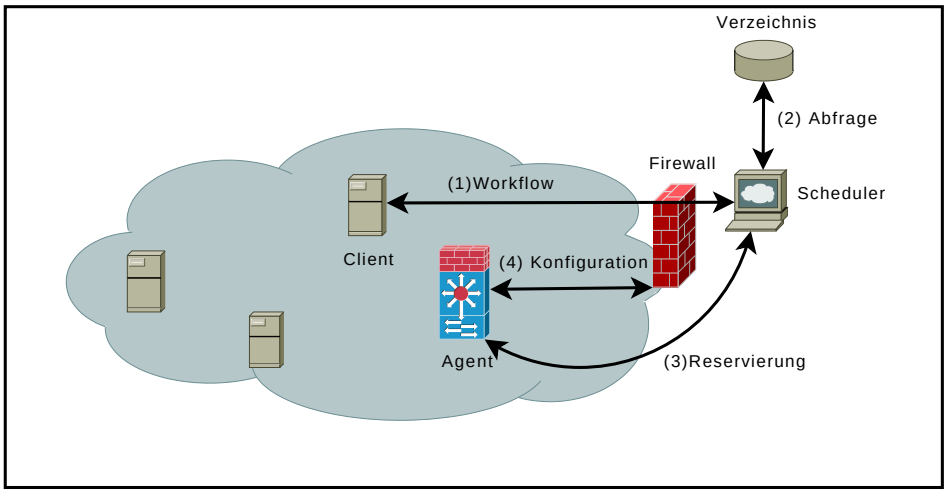

Abbildung 6.2: Architektur und Ablauf

Informationen über die Geräte im Grid in einer zentralen Datenbank vor, in der alle Systeme und die von den Systemen angebotenen Dienste verzeichnet sind. Die Grid-Komponenten können hier z.B. URLs von anderen Systemen erfragen, mit denen sie kommunizieren wollen.

Die Agenten sind eine neue Art von Grid-Komponenten, die in der Datenbank eingetragen werden müssen. Dazu werden die URL und die vom Agenten angebotenen Dienste erfasst. Die Dienste,die ein Agent bereit stellt, wurden bereits in Kapitel 3.4 vorgestellt.

Zu der Art der Komponente und den angebotenen Diensten kommt noch die Information über die Site, die der Agent betreut. Dies ist immer die Site, in der er selber steht. Einem Agenten, der in einem fremden Netz steht würde man keinen Zugriff auf die eigene Firewall geben. Die Site einer Komponente ist Teil der WebService-URL, die auf deren Dienste verweist. Diese Information ist also bereits vorhanden, sowohl für den Agenten, als auch für die Systeme, die eine Verbindungsfreigabe brauchen.

Der MSS kann nun, wenn er die beiden Endpunkte eines Transfers erhält, die beteiligten Sites und die zuständigen Agenten in der Datenbank ermitteln und bei diesen die nötige Freigabe beantragen.

6.1.3 Ablauf einer Reservierung

Die einzelnen Schritte einer Reservierung von Verbindungsfreigaben sollen hier nochmals zu einem kompletten Ablauf zusammengefasst werden. Einen Überblick über das Verfahren gibt Abbildung 6.2. Initiiert wird der Prozess durch die Reservierungsanfrage des Clients an den MSS (1). Dieser erhält vom Client den erstellten Workflow, der unter anderem Dateitransfers zwischen Systemen im Grid enthält. Hierfür müssen Netzwerk-Ressourcen, also auch Verbindungsfreigaben, reserviert werden. Der MSS extrahiert die URLs der beiden Systeme aus den Workflow-Informationen und erhält so auch die Namen der Sites, in denen sich die Systeme befinden. Mit Hilfe der Sites kann der MSS in der Datenbank nach dem Agenten suchen (2), der für diese Site zuständig ist. Diesen Agent spricht der MMS dann an, und handelt mit ihm über das WS-AGreement Protokoll die Reservierung aus (3). Welche Reservierung er genau braucht, ist aus den Informationen über den Dateitransfer im Workflow ersichtlich.

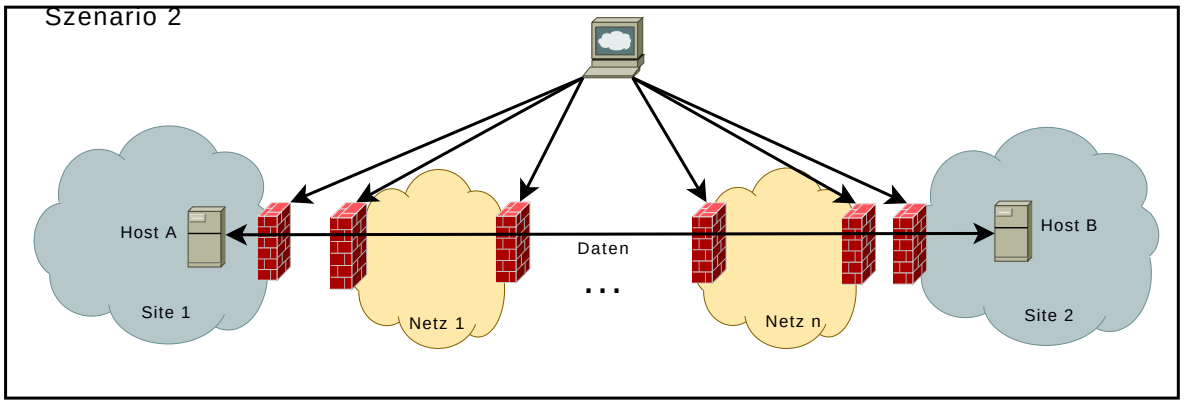

Abbildung 6.3: Szenario 2

Ergebnis der Verhandlung ist ein Agreement Objekt, welches vom Agenten verwaltet wird. Der Agent sendet Konfigurationskommandos an die Firewall (4), sobald die Reservierung erfolgreich vereinbart wurde. Die Freischaltung ist damit erfolgt. Der MSS benachrichtigt nach erfolgreicher Reservierung sämtlicher Ressourcen den Client, welcher dann den Job starten kann. Sollten andere Ressourcen nicht reserviert werden können, kann der MSS das Agreement vorzeitig terminieren. Der Agent sendet dann Konfigurationskommandos zur Löschung der Regeln an die Firewall. Wenn der im Agreement vereinbarte Zeitpunkt zur Aufhebung der Freigabe erreicht ist, löscht der Agent ebenfalls die betreffende Konfiguration. Das Agreement Objekt wird dann zerstört. Sämtliche Kommunikation der einzelnen Komponenten geschieht mit den Standardmitteln der Grid-Middleware, die auch Authentifizierung, Autorisierung und Verschlüsselung bereitstellt.

6.2 Umfassenderes Szenario

Das bisher beschriebene Szenario ist in einem Punkt sehr einfach gehalten: es beschreibt die Weiterleitung der Daten durch das allgemeine INTERNET. In realen Grids, besonders im Bereich der öffentlichen Forschung, sind aber häufig spezielle Netze im Einsatz, die extra für die Kommunikation zwischen wissenschaftlichen Rechnern entworfen wurden. Diese eigenständigen Netze werden von internationalen Projekten oder nationalen Verbänden betrieben und sind wiederum untereinander verbunden. Dabei werden verschiedene Techniken eingesetzt. Jedes Netz hat seine eigene Verwaltung, eigene Mechanismen zur Authentifizierung und Autorisierung und andere Methoden zur Konfiguration.

Wenn in einem Grid mehrere solcher Spezialnetze genutzt werden, dann wird die Freischaltung der Verbindungen komplizierter. Es sind mehr Freischaltungen nötig, weil bei jedem Netz, durch das der Datenstrom fließt, möglicherweise zwei Firewalls an den beiden Grenzen des Netzes passiert werden. Zudem muss festgestellt werden, welche Netze überhaupt genutzt werden. Das ist bei aufwändigen Topologien nicht mehr trivial aus den Adressen der Endpunkte ersichtlich. Abbildung 6.3 zeigt grob das zu behandelnde Problem.

Die Freischaltung der Verbindungen ist aber nicht das einzige Problem, das sich in solchen Szenarien ergibt. Auch bei der dynamischen Konfiguration der Kommunikationspfade, der Reservierung von Bandbreiten und dem Accounting sind erweiterte Mechanismen ge-

fragt. Daher ist eine umfassende Lösung nötig. Ein Konzept für eine solche Lösung erarbeitet z.B. das Phosphorus Projekt [10]. Der dort entwickelte Systemaufbau besteht aus mehreren Schichten, die im folgenden kurz vorgestellt werden.

Entwicklungsstufe 1

Sevice Plane Die Service Plane besteht aus der Grid-Middleware mit ihren diversen Service-Komponenten. Einer dieser Komponenten ist ein sogenannter co-allocation Service, also der bereits beschriebene Grid-Scheduler, der die Reservierung der diversen Ressourcen übernimmt. Der Grid-Scheduler reserviert sowohl die Rechner-Ressourcen wie Speicherplatz und CPU-Zeit als auch die Netzwerk-Ressourcen wie z.B. Bandbreite. Für die Netzwerk-Ressourcen kontaktiert der Scheduler die Network Resource Provisioning Systeme (NRPS) der benötigten Netze.

Network Resource Provisioning Plane Hier befinden sich die NRPS. Sie verwalten die Ressourcenvergabe ihres jeweiligen Netzes. Es gibt verschiedene Produkte für diese Aufgabe, die alle mit dem Scheduler zusammenarbeiten müssen. Die NRPS nehmen Reservierungen entgegen und übermitteln entsprechende Anweisungen an die Control Plane, welche die Reservierung technisch umsetzt. Auch untereinander tauschen die NRPS Informationen aus, um Reservierungen über mehrere autonome Systeme hinweg zu ermöglichen.

Control Plane Diese Schicht ist für die technische Umsetzung der Reservierungen zuständig. Sie besteht aus einer Kontrollschicht für das Generalized Multi Protocol Label Switching (GMPLS). Dieses Verfahren erlaubt die automatisierte Schaltung von Transportpfaden mit entsprechenden Dienstgarantien durch ein Netzwerk.

Resourcen Hierzu zählen die Netzwerk-Ressourcen, die über die beschriebenen Schichten reserviert werden, aber auch die traditionellen Ressourcen wie CPU-Zeit und Speicher. Diese werden in Entwicklungsstufe 1 unter Umgehung der anderen Schichten direkt vom Scheduler angesprochen.

Entwicklungsstufe 2

Eine vollständige Integration von System- und Netzwerk-Ressourcen wurde in Entwicklungsstufe 2 des Projektes umgesetzt. Die System-Ressourcen werden nicht mehr direkt vom Scheduler angesprochen. Anfragen hierzu werden ebenfalls über die NRPS- und Control-Plane gesendet. Dazu ist eine Erweiterung der Funktionalität der NRPS nötig und das GMPLS muss sich der System-Ressourcen bewusst werden. Als Ergebnis entstand das Grid Enabled GMPLS (G^2MPLS).

In diese fertige Struktur kann auch die Freischaltung von Verbindungen integriert werden. Abbildung 6.4 zeigt dabei die Zuordnung der einzelnen Komponenten zu den Ebenen. Der Firewall-Agent ist dabei auf der selben Ebene angesiedelt wie das NRPS. Durch entsprechende Anpassung kann das NRPS diese Funktionalität auch direkt übernehmen. Angesprochen wird der Agent wie im Konzept vorgesehen vom Scheduler auf der nächst höheren Ebene. Anstatt dann ein GMPLS-System auf der Control-Plane anzusprechen wird dort die Firewall ensprechend konfiguriert. Diese filtert den Netzwerk-Verkehr dann entsprechend der Konfiguration. Auch hier ist eine Eingliederung der Firewall-Konfiguration in das ohnehin für allgemeine Ressourcen erweiterte G^2MPLS naheliegend.

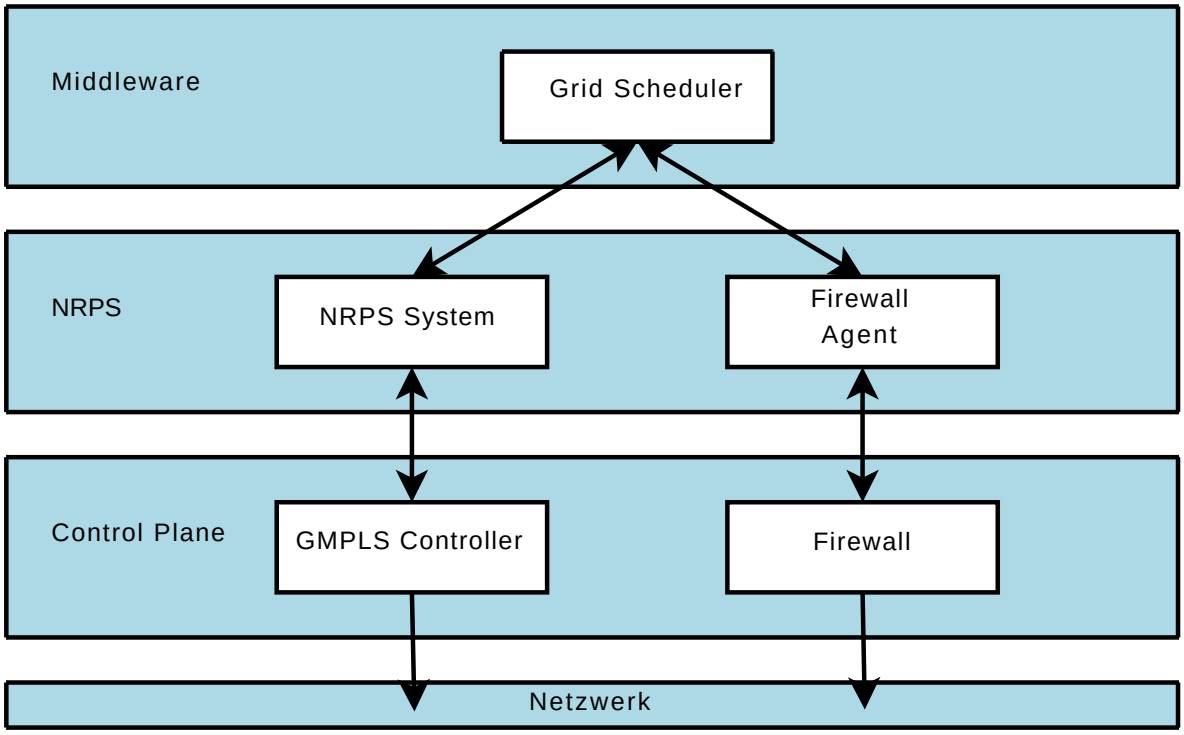

Abbildung 6.4: Architektur

Die bisherige Funktionalität ist mit geringem Aufwand erweiterbar, da Pfade und Randpunkte im Netz auch bisher schon ermittelt und konfiguriert werden mussten.

Die Autorisierung der Teilnehmer spielt in einem solchen Umfeld eine größere Rolle als im einfachen Szenario. Die Verwaltung der beschriebenen *assertions* ist aufwändiger, da hier erstmals viele verschiedene Parteien auf verschiedenen technischen Ebenen zusammen arbeiten. Die Sicherheitsattribute der Teilnehmer und die hinterlegten Autorisierungsentscheidungen müssen für alle Parteien im Grid verständlich sein und von ihnen akzeptiert werden. Dazu ist eine übergreifende Sicherheitspolitik für die gesamte VO nötig, die auf der lokalen Sicherheitspolitik der einzelnen realen Organisationen aufbaut.

Kapitel 7

Sicherheit

Ziel dieser Arbeit ist unter anderem, ein Verfahren zu entwickeln, das eine hohe Netzwerksicherheit ermöglicht. Eine genaue sicherheitskritische Betrachtung des Verfahrens ist daher notwendig. Dabei sind verschiedene Bereiche zu untersuchen. Verhandlungen über die Freischaltung finden auf Grid-Ebene in der Middleware statt. Dabei ist besonders die Autorisierung zu beachten. Die technische Freischaltung geschieht anschließend zwischen Agent und Firewall. Die einzelnen Vorgänge werden hier gesondert betrachtet. Dazu werden jeweils wieder die üblichen Begriffe Verfügbarkeit, Integrität, Vertraulichkeit und Widerspruchsfreiheit herangezogen, die bereits in Kapitel 2.1.1 eingeführt wurden.

7.1 Middleware

Die Grid-Middleware regelt die gesamte Kommunikation zwischen den Komponenten des Grid. Dazu bietet sie bereits Mechanismen zur Autorisierung und Authentifizierung von Systemen und Nutzern. Dabei kommen normalerweise Zertifikate zum Einsatz. Die Nachrichten zwischen den Grid-Komponenten können zudem verschlüsselt werden. Diese Mechanismen bieten bei richtiger Konfiguration alle Eigenschaften, die für einen sicheren Betrieb nötig sind:

Vertraulichkeit: Informationen über die im eigenen Netz vorhandenen Systeme, ihre Adressen und benutzte Ports können von Angreifern missbraucht werden. Sie sind daher vertraulich zu behandeln. Das gilt auch für Informationen über die Benutzer und ihre Berechtigungen. Vertraulichkeit wird hier durch die Verschlüsselung der Kontrollverbindungen im Grid sichergestellt. Dass die Daten nur an den berechtigten Empfänger gesendet werden ist durch die Authentifizierung der Teilnehmer mittels Zertifikaten gewährleistet.

Integrität: Wichtig ist, dass die Freischaltungsdaten nicht verfälscht oder manipuliert werden, damit keine falschen Verbindungen freigegeben werden. Dies könnte Angreifern den Zugriff auf das eigene Netzwerk ermöglichen. Sichergestellt wird dies wiederum durch die Kombination von zuverlässigen Transportprotokollen, Verschlüsselung und Authentifizierung mittels Zertifikaten.

Verfügbarkeit: Hier wird die Verfügbarkeit der Middleware-Dienste betrachtet (zur Verfügbarkeit des Netzes s.u.). Damit die Middleware ihre Aufgaben erfolgreich durchführen kann ist es nötig, dass vereinbarte Freischaltungen auch umgesetzt werden. Dies stellt die Factory sicher, indem sie das Agreement-Objekt erst erstellt, wenn die Freischaltung durchgeführt wurde. Der Agent als Teil des Grids muss ständig verfügbar sein, damit die anderen Komponenten arbeiten können.

Widerspruchsfreiheit: Die Widerspruchsfreiheit kann durch die Verwendung von Zertifkaten erreicht werden. Nachrichten können mit diesen unterschrieben werden, so das sie einem Absender fest zugeordnet werden können. Durch Archivierung der wesentlichen Nachrichten durch die Factory ist genau nachvollziehbar, wer wann welche Freischaltung beantragt hat. Zu erweiterten Mechanismen hierfür, siehe Kapitel 7.2.

Zusammenfassend kann man sagen, dass heutige Middleware mit ihren modernen Methoden zur Autorisierung, Authentifizierung und Verschlüsselung einen ausreichend sicheren Rahmen zur Aushandlung von Freischaltungen bieten.

7.2 Autorisierung

Neben den technischen Voraussetzungen zur sicheren Durchführung der Freischaltungen muss die Autorisierung einzelner Nutzer genau betrachtet werden. Die Grid-Middleware stellt nur sicher, dass es sich bei dem Antragsteller um einen ordentlichen Nutzer des Grids handelt. Die Freischaltung von Transportverbindungen bedarf aber für einen sicheren Betrieb einer viel genaueren Unterscheidung der Nutzer. Bestimmten Gruppen oder Einzelpersonen soll möglicherweise Zugang zu bestimmten Netzen gewährt werden. Daher muss konfigurierbar sein, welche Nutzer zu welchen Zeitpunkten Freischaltungen beantragen dürfen und auf welche Systeme bzw. Netze dabei zugegriffen werden darf. Als Lösung hierfür wurde der SAML-Standard genannt, mit dessen Hilfe detailierte Informationen zur Autorisierung der einzelnen Nutzer verwaltet und ausgetauscht werden können.

Alle nötigen Autorisierungsinformationen können in den SDTs verpackt werden. Somit können die einzelnen Netzwerkadministratoren in ihren Factories genau konfigurieren, wer welche Rechte hat. Das Grid verwaltet die Nutzerinformationen, die der Scheduler dann in sein Angebot integriert. Dieses Konzept bedarf der entsprechenden Unterstützung durch die Grid-Middleware und ist daher nicht nur durch Anpassungen des domain specific Teils realisierbar.

Besonders in größeren VOs müssen das Verhalten der Middleware und der Inhalt der Sicherheitsattribute die Sicherheitsrichtlinien der realen Organisationen erfüllen. Die nötige Grid-weite Verwaltung von Identitäten und Rechten kann durch den für SAML vorgesehenen *Identity Provider* realisiert weden. Eine solche zentrale Stelle bietet dann alle nötigen Informationen für die gesamte VO. Da jede VO einen eigenen *Identity Provider* hat, kann eine reale Person in verschiedenen VOs verschieden behandelt werden.

7.3 Freischaltung

Neben der Kommunikation auf Grid-Ebene zwischen Agent und Scheduler wird auch zwischen dem Agenten und der Firewall kommuniziert. Hier muss ein Verfahren eingesetzt werden, das die jeweilige Firewall zur Fernkonfiguration bereit stellt. Dieses Verfahren sollte die folgenden Eigenschaften bieten:

Verschlüsselung Die wichtigste Funktion des Verfahren ist die Verschlüsselung aller übertragenen Daten. Starke symmetrische Verschlüsselung gewährt sowohl Vertraulichkeit als auch Integrität der Daten. Der Schlüssel kann z.B. während des asymetrisch verschlüsselten Logins vereinbart werden.

Schlüsselbasierter Login Bei gängigen Verfahren wie ssh kann eine Anmeldung entweder mit Benutzername und Passwort erfolgen, oder über ein asymetrisches Schlüsselpaar. Benutzername und Passwort müssten entweder beim Agenten in Klartext hinterlegt werden, was nicht dem Sicherheitsanspruch genügt oder von Hand eingegeben werden, wodurch das Gesamtverfahren nicht mehr automatisch wäre. Schlüsselpaare sind dagegen eine sichere und flexible Möglichkeit zur Anmeldung.

Das genaue Verfahren in diesem Bereich ist also vom Typ der Firewall abhängig. Im Allgemeinen bieten Firewalls als sicherheitskritische Systeme gut durchdachte Verfahren an, so das für ausreichende Sicherheit gesorgt ist.

Widerspruchsfreiheit ist in diesem Fall gegeben, da Agent und Firewall von der selben Organisation betrieben werden und es zu jeder Firewall ohnehin nur einen Agenten geben soll.

Es genügt somit für die Widerspruchsfreiheit, dass sowohl der Agent, als auch die Firewall ein angemessenes Logging der durchgeführten Aktionen bereitstellen.

7.4 Netzwerk und Firewall

Die Verfügbarkeit des Netzwerks wird durch die Freischaltungen nicht gefährdet, da die Konnektivität durch zusätzliche Freischaltungen nicht eingeschränkt, sondern nur erweitert wird. Diese Annahme setzt vorraus, das die Software korrekt implementiert ist und es somit nicht zu falschen Konfigurationen kommen kann. Firewalls können üblicherweise im laufenden Betrieb konfiguriert werden, so das die ständige Freischaltung von Verbindungen die Verfügbarkeit der Firewall nicht einschränkt. Ist die Firewall für den Agenten nicht mehr erreichbar, dann können keine weiteren Freischaltungen erfolgen. Eine nicht ansprechbare Firewall stellt aber onehin ein gravierendes Problem dar, das unabhängig vom Grid-Betrieb zu vermeiden ist.

Zudem müssen die Regeln, basierend auf der Auswertungsstrategie der Firewall, an der richtigen Stelle im Regelwerk eingefügt werden, damit sie auch wirksam werden. Das genaue Vorgehen ist hier Firewall-spezifisch.

Kapitel 8

Bewertung und Ergebnisse

8.1 Bewertung und Vergleich

Abschließend soll das vorgestellte Konzept mit bereits existierenden Verfahren verglichen werden. Dabei ist einerseits die Konfigurationsmethode der Firewall zu beachten und andererseits das eigentliche Dateitransferverfahren. In Kapitel 2.2 wurden bereits die wesentlichen Verfahren und ihre wichtigsten Eigenschaften vorgestellt.

Zuerst werden die Konfigurationsverfahren verglichen:

Manuelle Konfiguration Die klassische Vorgehensweise zur Konfiguration der Firewall besteht aus dem Festlegen von Regeln durch einen Administrator. Dieses Verfahren hat den Nachteil, das es einen hohen, nicht vertretbaren Aufwand erfordert, wenn sich ständig Änderungen am Regelwerk ergeben. Dies ist bei Betrieb eines Grids der Fall und wird normalerweise dadurch gelöst, dass Regeln für größere Bereiche erstellt werden, die das Grid nicht beeinträchtigen, aber auch unnötige Kommunikation erlauben. Manuelle Konfiguration ist also entweder sehr aufwändig und nicht zeitnah, oder sie führt unter den in einem Grid gegebenen Vorausetzungen zu unsicherer Konfiguration des Netzwerks.

UDP Hole Punching Das UDP Hole Punching erlaubt dynamische Konfiguration der Firewall durch die Systeme im eigenen Netz. Der manuelle Teil des Regelwerks erlaubt hierbei nur ausgehenden UDP Verkehr, was in den meisten Fällen ausreichend sicher ist. Dieses Verfahren benötigt ein passendes Transferverfahren auf Basis von UDP. Es ist sicher, aber nicht so flexibel wie das neue Verfahren.

Aktive Freischaltung Die aktive Freischaltung von Verbindungen durch einen Agenten wurde in dieser Arbeit beschrieben. Sie integriert den Vorgang des Freischaltens in die Grid-Middleware und nutzt dadurch die umfangreichen Möglichkeiten zur Authentifizierung, Autorisierung und Verschlüsselung. Dadurch ist das Verfahren besser zu verwalten und kontrollieren als das UDP Hole Punching. Selbst ausgehender Verkehr muss hier nicht statisch erlaubt werden. Zudem ist jedes beliebige Verfahren zum Transfer der Daten einsetzbar. Diese Methode bietet hohen Schutz und maximale Flexibilität.

Die folgenden Transferverfahren können genutzt werden:

Tunneling Alle Tunneling-Verfahren haben gemeinsam, dass sie keinen direkten Datenaustausch zwischen den Systemen benötigen. Daher ist eine dynamische Konfiguration nicht nötig. Das statische Regelwerk ist einfach und übersichtlich. Aus Sicht der Netzwerksicherheit ist dies der ideale Fall. Allerdings ist Tunneling langsam, weil es aufwändiges Multiplexen der Verbindungen erfordert und die Tunnel-Server Bottlenecks darstellen. Zudem stellen die Server Single-Point-of-Failures dar. Direkte Verfahren sind bei ausreichender Netzwerksicherheit zu bevorzugen.

GridFTP GridFTP ist durch die direkten parallelen Verbindungen sehr schnell. Bei manueller Firewall-Konfiguration benötigt es aber ein sehr offenes und damit unsicheres Regelwerk. Mit UDP Hole Punching ist GridFTP nicht kombinierbar, da es auf TCP aufbaut. Die aktive Freischaltung der GridFTP Verbindungen über einen Agenten stellt aber eine sichere Möglichkeit zum Betrieb von GridFTP dar. In dieser Kombination ist das Verfahren eine gute Lösung für Grids.

UDT UDT basiert auf UDP und kann dadurch mit UDP Hole Punching genutzt werden. Auch die Verwendung mit aktiver Freischaltung ist möglich. UDT ist ein sehr schnelles Verfahren, das mit beiden möglichen Konfigurationsmethoden einen sicheren Betrieb erlaubt. Es stellt somit eine gleichwertige Alternative zu GridFTP dar.

Insgesamt zeigt sich, dass die aktive Freischaltung die flexibelste Lösung zur dynamischen Konfiguration der Firewall darstellt. Sie erlaubt die Nutzung aller gängigen Transferverfahren. Zudem ist genaue Kontrolle der Freigaben möglich.

8.2 Ergebnisse

Ziel der Arbeit war es, ein sicheres und leicht zu konfigurierendes Konzept zu finden, das sich gut mit der Architektur eines Grid vereinbaren lässt. Die aktive Freischaltung mittels eines Agent ist sehr gut in eine Grid-Middleware integrierbar. Sie bietet Sicherheit bei maximaler Flexibilität und hoher Kontrolle. Die Mechanismen des Grid zur Authentifizierung und Autorisierung können genutzt werden. Das dynamische Freischalten geschieht zeitlich beschränkt und beliebig präzise. Es können alle denkbaren Transport-Verfahren genutzt werden.

Die Implementierung des Konzepts ist allerdings aufwändiger als die Nutzung von UDP Hole Punching. Die Grid-Middleware muss angepasst werden und das gesamte Scheduling sowie die Methoden zum Datei-Transfer müssen die nötige Freischaltung beachten. Die Verwendung von Standards wie WS-Agreement erleicht diese Aufgabe allerdings. Wenn erste Grid-Middleware diesen Standard umsetzt und vollständig integriert, dann kann auch das vorgestellte Konzept als praxistaugliches Verfahren implementiert werden.

8.3 Ausblick

Im Bereich der dynamischen Konfiguration von Firewalls werden zur Zeit verschiedene Lösungen entwickelt. Das Verfahren des UDP Hole Punching in Verbindung mit UDT wurde als erstes umgesetzt, weil es schnell und einfach zu implementieren ist und sofort in der Praxis eingesetzt werden kann. Aktive Freischaltung ist ein mächtigeres, aber auch komplizierteres Konzept, das erst mittelfristig in die verbreitete Grid-Middleware integriert werden kann.

Weitere Ansätze zur dynamischen Konfiguration setzen auf Protokolle, die von der Firewall dynamisch analysiert werden. Dies betrifft z.B. FTP oder SIP, die in einer Kontrollverbindung die genauen Parameter für Datenverbindungen aushandeln. Die Firewall liest diese Informationen mit und erlaubt die Datenverbindungen dynamisch. Aktuelle Forschung versucht, diese bekannten und standardisierten Protokolle zu nutzen. Alternativ können neue Protokolle definiert werden, die auf schnelle Transfers im Grid spezialisiert sind. Diese müssen aber von den Firewalls erst unterstützt werden, wenn sie nicht auf standardisierten Mechanismen aufbauen.

In Zukunft wird es also eine Vielzahl von Möglichkeiten geben, die bisherigen Probleme bei der Vereinbarung von Netzwerksicherheit und Grid-Betrieb zu umgehen.

Literaturverzeichnis

[1] O. Wäldrich, P. Wieder, W. Ziegler, *A Meta-Scheduling Service for Co-allocating Arbitrary Types of Resources*, CoreGRID Technical Report Number TR-0010, 2005

[2] A. Pichot, P. Wieder, O. Wäldrich, W. Ziegler, *Dynamic SLA-negotiation based on WS-Agreement*, CoreGRID Technical Report Number TR-0082, 2007

[3] W. Ziegler, P. Wieder, Dominic Battre, *Extending WS-Agreement for dynamic negotiation of Service Level Agreements*, CoreGRID Technical Report Number TR-0172, 2008

[4] T. Oistrez, *Ein zuverlässiger und schneller Dateitransfer mit dynamischer Firewall-Konfiguration für Grid-Systeme*, Forschungszentrum Jülich, FZJ-JSC-IB-2008-02, 2008

[5] OpenGridForum (GRAAP WG), *Web Services Agreement Specification (WS-Agreement)*, 2007

[6] OpenGridForum , *Job Submission Description Language (JSDL) Specification*, 2005

[7] OASIS, *Security Assertion Markup Language, Specification*, http://saml.xml.org/saml-specifications

[8] R. Niederberger, T. Oistrez, *Firewalls and Grids - Update on UDP Hole Punching*, Open Grid Forum, OGF23 Barcelona 2008

[9] R. Niederberger, I. Monga, T. Metsch, *Firewall Virtualization for Grid Applications*, Open Grid Forum, OGF23 Barcelona 2008

[10] S. Figuerola, N. Ciulli, M. De Leenheer, Y. Demchenko, W. Ziegler, A. Binczewski, *PHOSPHORUS: Single-step on-demand services across multi-domain networks for e-science*, the PHOSPHORUS consortium, 2007

[11] R. Russell, *Linux 2.4 Packet Filtering HOWTO*, Version 1.0.2, www.netfilter.org, 2000

[12] World Wide Web Consortium - *Web Services Activity*, www.w3.org/2002/ws/

[13] GridFTP Project, www.globus.org/grid_software/data/gridftp.php

[14] Fraunhofer SCAI, *wsag4j*, http://packcs-e0.scai.fraunhofer.de/wsag4j

[15] UNICORE - Uniform Interface to Computing Resources, www.unicore.eu

[16] SETI@HOME - Search for Extraterrestrial Intelligence, setiathome.berkeley.edu

[17] FOLDING@HOME - protein folding, folding.stanford.edu

[18] IPv6 - Internet Protocol Version 6, www.ipv6.org

Anhang A

XML Schema ConnectionDuration

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://ks.jsc.fzj.de/schema/ConnectionDuration"
            xmlns:tns="http://ks.jsc.fzj.de/schema/ConnectionDuration"
            elementFormDefault="qualified">

    <xsd:complexType name="ConnectionDurationType">
        <xsd:sequence>
            <xsd:element name="BeginEstablishment"
                         type="xsd:dateTime">
            </xsd:element>
            <xsd:element name="EndEstablishment"
                         type="xsd:dateTime">
            </xsd:element>
            <xsd:element name="EndConnection"
                         type="xsd:dateTime" minOccurs="0">
            </xsd:element>
        </xsd:sequence>

    </xsd:complexType>
    <xsd:element name="ConnectionDuration"
                 type="tns:ConnectionDurationType">
    </xsd:element>
</xsd:schema>
```

Anhang B

XML Schema ConnectionDescription

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://ks.jsc.fzj.de/schema/ConnectionDescription"
            xmlns:tns="http://ks.jsc.fzj.de/schema/ConnectionDescription"
            elementFormDefault="qualified">

    <xsd:simpleType name="ConnectionPortNumberType">
        <xsd:restriction base="xsd:integer">
            <xsd:minInclusive value="1"/>
            <xsd:maxInclusive value="65535"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="ConnectionTransportProtocolType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="UDP"/>
            <xsd:enumeration value="TCP"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="ConnectionAddressType">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="ConnectionNetworkProtocolType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="IPv4"/>
            <xsd:enumeration value="IPv6"/>
        </xsd:restriction>
    </xsd:simpleType>
```

```
    <xsd:complexType name="ConnectionDescriptionType">
        <xsd:sequence>
            <xsd:element name="NetworkProtocol"
                         type="tns:ConnectionNetworkProtocolType">
            </xsd:element>
            <xsd:element name="TransportProtocol"
                         type="tns:ConnectionTransportProtocolType">
            </xsd:element>
            <xsd:element name="SourceAddress"
                         type="tns:ConnectionAddressType">
            </xsd:element>
            <xsd:element name="DestinationAddress"
                         type="tns:ConnectionAddressType">
            </xsd:element>
            <xsd:element name="SourcePortRangeStart"
                         type="tns:ConnectionPortNumberType">
            </xsd:element>
            <xsd:element name="SourcePortRangeEnd"
                         type="tns:ConnectionPortNumberType">
            </xsd:element>
            <xsd:element name="DestinationPortRangeStart"
                         type="tns:ConnectionPortNumberType">
            </xsd:element>
            <xsd:element name="DestinationPortRangeEnd"
                         type="tns:ConnectionPortNumberType">
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="ConnectionDescription"
                 type="tns:ConnectionDescriptionType">
    </xsd:element>
</xsd:schema>
```

Jül-4304
Juli 2009
ISSN 0944-2952